Contrôle des communications dans les machines parallèles à mémoire distribuée : contribution au routage automatique des messages

Leon Mugwaneza

▶ **To cite this version:**

Leon Mugwaneza. Contrôle des communications dans les machines parallèles à mémoire distribuée : contribution au routage automatique des messages. Réseaux et télécommunications [cs.NI]. Institut National Polytechnique de Grenoble - INPG, 1993. Français. NNT : . tel-00005138

HAL Id: tel-00005138

https://theses.hal.science/tel-00005138

Submitted on 26 Feb 2004

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

THESE

présentée par

Léon MUGWANEZA

pour obtenir le grade de **DOCTEUR**
DE L'INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

(Arrêté ministériel du 30 mars 1992)

Spécialité : **Informatique**

Contrôle des Communications dans les Machines Parallèles à Mémoire Distribuée

Contribution au routage automatique des messages

Date de soutenance : 24 Novembre 1993

Composition du jury :

Président :	Guy	MAZARE
Rapporteurs :	Claude	GIRAULT
	Alan	KNOWLES
Examinateurs :	Traian	MUNTEAN
	Ibrahima	SAKHO
	Denis	TRYSTRAM

Thèse préparée au sein du Laboratoire de Génie Informatique

Remerciements

C'est avec grand plaisir que je remercie ceux grâce à qui cette thèse a pu voir le jour.

Guy Mazaré, directeur de l'ENSIMAG, pour l'honneur qu'il m'a fait en acceptant de présider mon jury de thèse.

Claude Girault, professeur à l'univerisité Paris VI, et Alan Knowles, senior lecturer à l'université de Manchester, pour avoir accepté de rapporter sur mon travail et pour leurs remarques.

Denis Trystram, professeur à l'INPG, pour l'intérêt qu'il a manifesté pour mon travail en acceptant de participer au jury.

Traian Muntean, mon directeur de thèse, qui m'a accueilli dans son groupe "Sympa ", et a su me procurer les moyens et un cadre pour m'exprimer.

Ibrahima Sakho, ingénieur de recherches à l'école des Mines de Saint-Etienne, pour l'amitié qu'il m'a toujours temoignée, pour les conseils (aussi bien dans mon travail qu' en dehors) qu'il a toujours prodigués à son cadet; bref, pour tout ce que nous avons pu faire ensemble.

Les "jeunes" chercheurs de l'équipe Sympa du Laboratoire de Génie Informatique, pour leur aide à toutes les étapes de ma vie de thésard. Merci donc à Yves, Nestor, Philippe(s), François et Ghazali, pour les moins jeunes. Merci à Alba, Leïla, Mihaela, Harold, et "mon petit Robert", pour les plus jeunes, merci à Ahmed qui a partagé mon intimité dans le bureau 45M.

J'ai aussi apprécié l'ambiance au "RDC Maths pures". Hélène, Martine, Fethi et Jérôme y sont pour beaucoup, merci à eux.

Enfin, je ne saurais clore cette liste, sans remercier tous mes amis chez qui j'ai pu trouvé encouragements et reconfort. Je tiens notamment à exprimer toute ma reconnaissance à la famille Evêque, et à mon "frère blanc" Philippe, pour m'avoir permis d'être en famille à des milliers de kilomètres de ma colline natale.

Résumé

Cette thèse traite d'un ensemble de problèmes liés à l'acheminement des messages dans les machines parallèles à mémoire distribuée. L'accent est mis sur des solutions extensibles qui nécessitent un nombre de ressources indépendant de la taille de la machine.

A travers l'exemple des machines Supernodes (dont les processeurs sont interconnectés par un réseau de Clos 3-étages) nous montrons que l'acheminement des messages par reconfiguration dynamique est difficilement envisageable dans des machines de grande taille. Nous nous intéressons ensuite au routage des messages dans des réseaux à topologie quelconque, et proposons une nouvelle méthode de génération de fonctions de routage sans interblocage.

La nouvelle génération des machines parallèles intègre de plus en plus de fonctions dans le matériel, notamment le routage des messages. Pour que cette intégration soit la plus efficace possible, des méthodes nouvelles de représentation compacte de l'information de routage sont nécessaires.

Santoro et Khatib ont proposé une méthode, le routage par intervalles, bien adaptée aux réseaux généraux. La deuxième partie de cette thèse s'inscrit dans la continuité de ce type de travail et propose de nouvelles méthodes de génération de fonctions de routage par intervalles. Deux cas sont considérés : le tore, et les réseaux généraux. Nous insistons plus particulièrement sur des solutions sans interblocage, caractéristique rarement prise en compte. De plus dans le cas du tore, les longueurs des chemins sont proches des optima.

Enfin, nous proposons une extension de la notion de routage par intervalles, le schéma d'étiquetage étendu (SEE), qui permet de représenter un spectre plus large de fonctions de routage.

Mots clés : Architectures parallèles, systèmes parallèles, échange de messages, réseaux d'interconnexion, processeurs de communication, interblocage, routage, routage par intervalles.

Abstract

Massivelly parallel computers with thousand and more processors are considered one of today's promising technologies to achieve high performance computing. Such large-scale multiprocessors machines are usually organized as sets of nodes, where each node has its own processor and local memory, connected by some interconnection network. As generally nodes do not share memory, they communicate by passing messages through the network. In this dissertation, we adress the problem of messages routing in massivelly parallel computers. The stress is put on scalable algorithms which require an amount of resources independent of the network size and shape.

Through the example of the supernode architecture (dynamically reconfigurable networks of transputers) we show that the complexity of handling the message exchanges by dynamically connecting processors is high in large scale machines. Our study focusses then on the problem of deadlock-free routing in non regular networks and we propose a novel algorithm.

Recently the trend in parallel computer architectures is to offer hardware support for handling messages exchanges within nodes. To efficiently achieve this objective for massivelly parallel computers, new methods for compacting routing information on a node are required.

A technique well suited for non regular networks is the interval routing introduced by Santoro and Khatib. For this kind of methods we propose deadlock-free solutions for k-ary ncubes and general networks. For the k-ary ncube our method gives moreover nearly optimal paths.

Finally, we propose an original extension for interval labelling which needs routing tables of size $O(d^2)$ (where d is the number of neighbors) for a node. This extension allows to represent more routing functions than the original interval labelling.

Key words : Parallel architectures, parallel systems, message passing, interconnection networks, communication processors, deadlock, message routing, interval routing, interval labelling.

Chapitre 1

INTRODUCTION

1.1 Du séquentiel au parallèle

Depuis l'avènement des ordinateurs modernes une des principales motivations de la recherche en informatique est l'amélioration des performances afin de pouvoir exécuter les applications toujours plus vite. Les recherches ont été menées sur trois fronts :

- algorithmique : en recherchant les algorithmes les plus efficaces,
- compilation : en cherchant à faire des optimisations lors de la traduction des programmes (compilateurs optimiseurs), et
- matériel.

Augmenter les performances des machines au niveau matériel peut se faire de deux façons. Soit par l'utilisation de nouvelles technologies pour faire des circuits de plus en plus rapides, on améliore ainsi les performances d'un seul processeur (machine *séquentielle*). La deuxième solution consistant à utiliser plusieurs processeurs et diviser le problème à résoudre en plusieurs sous problèmes qui sont traités en *parallèle*.

C'est la première approche qui a longtemps été utilisée, mais les limites physiques d'une part, le rapport coût/performances de l'approche séquentielle et les applications demandeurs en puissances de calcul (mécaniques des fluides, prévisions météorologiques, etc.) d'autre part, ont fait naître un intérêt pour de nouvelles solutions moins chères et plus puissantes. Le parallélisme est alors une solution activement étudiée pour augmenter les performances des machines. Ainsi, d'abord dans les laboratoires de recherche, puis chez les industriels a t-on vu des machines parallèles construites. A. Trew et G. Wilson font une présentation assez complète de toutes les réalisations dans ce domaine [Trew91].

1.2 Les architectures parallèles

Parmi les diverses architectures parallèles proposées, nous considérons une classe d'architectures dites à *parallélisme massif* : les machines comportant un nombre important de nœuds de calcul autonomes ayant chacun un processeur et une mémoire locale privée. Ces nœuds sont interconnectés par un ensemble de liaisons point à point (*le réseau d'interconnexion*) et, en l'absence d'une mémoire commune, communiquent et se synchronisent par *échanges de messages*. De par le fait que chaque processeur est connecté à un petit nombre de voisins, ces machines sont facilement extensibles et permettent des configurations avec aujourd'hui des milliers de processeurs.

Les performances de ce type d'architectures sont tributaires de la vitesse des communications entre les processeurs, qui dépend, entre autres paramètres, de la topologie du réseau d'interconnexion choisie lors de la construction de la machine. On distingue les machines à réseau d'interconnexion fixe dont la topologie est fixée une fois pour toutes, et les machines dites reconfigurables dont le réseau d'interconnexion permet de réaliser par programme plusieurs types de topologies.

Machines à réseau d'interconnexion fixe

Ce furent les premières machines parallèles construites. Le réseau d'interconnexion idéal est sans conteste le réseau complètement connecté dans lequel chaque processeur est directement connecté à tous les autres. Ce réseau devient irréalisable dès qu'on dépasse la dizaine de processeurs. On se contente alors de réseaux où un processeur n'est relié qu'à un ensemble limité de voisins, la communication avec les autres processeurs se faisant par *routage* à travers des processeurs intermédiaires.
Les réseaux d'interconnexions fixes sont souvent classés par rapport à un ensemble de caractéristiques que sont [Wittie81, Agrawal86] :

- le diamètre du réseau
- la distance moyenne
- le nombre de liens de communication par processeur (degré)
- les algorithmes de routage qui doivent être les plus simples possibles et ne doivent pas requérir une connaissance totale du réseau.
- la tolérance aux pannes : il faut avoir au moins deux chemins différents entre chaque paire de processeurs. Cette "redondance " est utilisée en cas d'erreur
- la possibilité d'expansion, c'est à dire, pouvoir étendre le réseau sans "défaire" ce qui existe. Ceci suppose notamment que les composants de base doivent rester les mêmes.

Machines à réseau d'interconnexion programmable

Les machines à réseaux fixes ont des caractéristiques de communication qui peuvent convenir très bien à certaines applications et moins bien à d'autres. Un réseau d'interconnexion programmable (encore appelé dynamique ou reconfigurable), en permettant de réaliser plusieurs types de topologies, est plus souple et permet de prendre en compte les besoins des applications. Les réseaux d'interconnexion programmables ont d'abord été utilisés dans les multiprocesseurs à mémoire partagée. Mais ces dernières années on commence à les utiliser dans les machines sans mémoire commune.

Le réseau d'interconnexion programmable le plus simple est un commutateur matriciel (crossbar) permettant de relier N entrées à N sorties à l'aide de N^2 points de croisements (commutateurs élémentaires).
Le nombre de commutateurs élémentaires utilisés dans les crossbars les rend rapidement irréalisables lorsque on envisage des machines de tailles importantes. De plus, ces composants sont limités par le nombre de pattes qu'on peut mettre sur un seul circuit. La technologie actuelle ne permet que des crossbars dont la taille est de l'ordre 200×200. Pour des réseaux de tailles supérieures, on a recours à des réseaux multi-étages.

Les principales caractéristiques d'une machine à réseau d'interconnexion programmable sont :

- le degré des processeurs,
- la capacité du réseau d'interconnexion, c'est à dire le nombre de connexions qu'il peut réaliser sans conflit,
- la simplicité du contrôle, c'est à dire la possibilité de rajouter de nouvelles connexions en cours de fonctionnement,
- le délai de traversée du réseau.

1.3 Programmation et contrôle des machines parallèles

Même si beaucoup de machines parallèles ont été construites, leur programmation reste difficile. Elles partagent toutes la caractéristique suivante : peu ou pas de fonctions systèmes offrant une virtualisation de la machine.

Le système d'exploitation d'une machine séquentielle a pour principal objectif de définir une machine virtuelle qui, non seulement étend l'architecture matérielle sous-jacente, mais aussi cache les spécificités de celle-ci qui sont contraignantes pour l'utilisateur. Une telle machine virtuelle doit notamment offrir un ensemble de services comme la gestion des ressources, l'ordonnancement, la tolérance aux pannes, etc.

Dans le cas d'une machine parallèle, la machine virtuelle définie par le système d'exploitation doit offrir, en plus des services cités ci-dessus, un support pour l'ordonnancement des tâches parallèles, la communication et synchronisation entre nœuds distants, la tolérance aux pannes des nœuds, etc.

L'utilisation des machines parallèles pose le problème du formalisme d'expression du parallélisme, c'est à dire le choix d'un langage de programmation, ainsi que celui de l'implantation du programme sur la machine parallèle. L.C. Chang et B. T. Smith font une classification des outils pour la programmation des machines parallèles dans [Chang90]. D'une manière générale deux optiques ont été suivies pour l'expression du parallélisme :
Le besoin de préserver l'investissement pousse à l'utilisation des langages classiques avec des compilateurs capables de détecter les traitements indépendants. Beaucoup de travaux ont été menés notamment sur la parallélisation automatique de FORTRAN [Kennedy80, Allen88, Cytron89, Tawbi89, Wolfe89, Blume92, Hall93]. Cette approche est toutefois limitée. En effet, il arrive souvent qu'un bon algorithme parallèle pour un problème donné soit substantiellement différent d'un algorithme séquentiel efficace pour le même problème[Anderson84] ; ceci explique notamment le grand intérêt porté sur l'algorithmique parallèle par de nombreuses équipes de recherche à travers le monde.

L'autre optique est l'utilisation des langages parallèles permettant à l'utilisateur d'exprimer le parallélisme. Ceci peut être fait implicitement, c'est le cas de la plupart des langages logiques, et fonctionnels [Kelly89, Foster90], ou explicitement, c'est le cas de la plupart des langages impératifs comme OCCAM [Occam2], ou d'autres langages parallèles issus des langages séquentiels avec des extensions ad hoc permettant d'exprimer l'indépendance des traitements (Fortran parallèle, C parallèle, etc.).

Quel que soit le mode d'expression du parallélisme utilisé, l'implantation d'un programme sur une machine parallèle pose des problèmes nouveaux. A la traditionnelle chaîne de traduction (compilation, édition des liens, chargement) sur une machine séquentielle viennent s'ajouter de nouvelles tâches dont dépendent les performances de l'application [Eudes87]. Nous en citons les trois plus importantes à notre point de vue :

- L'application doit d'abord être partagée en plusieurs tâches (processus) pouvant s'exécuter en parallèle : c'est le problème de *partitionnement*.
- Les divers processus composant l'application doivent être affectés aux divers processeurs de la machine : c'est le problème d'*allocation*.
- Comme en général les divers processus de l'application ne sont pas complètement indépendants, on doit s'occuper de la coordination de leurs exécutions respectives : c'est le problème du *contrôle des communications et synchronisations*.

1.3.1 Le problème de partitionnement

Le partitionnement d'un programme parallèle a pour but d'identifier et spécifier des parties du programme pouvant s'exécuter en parallèle, et d'extraire un certain nombre d'informations sur la structure du programme. Chaque partie de programme, que nous appelerons indifféremment tâche ou processus, s'exécute séquentielilement sur un processeur. Une tâche est souvent caractérisée par :

- Son temps d'exécution séquentiel, nous parlerons aussi de sa taille
- Les autres tâches avec lesquelles elle communique et la quantité d'informations échangées
- Les contraintes de précédence entre tâches qui définissent le parallélisme dans le programme partitionné.

Le partitionnement de programmes n'est pas un problème facile et peut dépendre de l'architecture cible. Exposer tout le parallélisme d'une application peut conduire à une situation où l'on a beaucoup de tâches de petite taille et un coût de communication entre les tâches relativement important. Suivant l'importance relative des coûts de communications et des coûts d'exécution sur une machine parallèle, on peut ne pas améliorer le temps d'exécution parallèle. Le temps d'exécution parallèle étant minimum pour une certaine *granularite* intermédiaire des partitions. Le choix de la granularité optimale doit résulter d'un compromis entre l'extraction d'un maximum de parallélisme et la minimisation des informations échangées entre les tâches. Le problème de partitionnement de programmes est NP-Complet [Sarkar89b].

1.3.1.1 Extraction des coûts de communication

L'évaluation quantitative des communications entre deux processus communicants n'est pas facile. Certaines communications sont non déterministes ou dépendent des entrées du programme. On cherchera donc une approximation qui doit être expressive, c'est à dire qui garde l'importance relative des coûts exacts. Plusieurs approximations peuvent être utilisées :

Nombre de canaux de communication

Dans une application où le volume des données échangées sur les divers canaux de communication est identique (c'est le cas par exemple de quelques applications dans le calcul scientifique), le nombre de canaux de communication peut servir comme mesure de la quantité de communications entre 2 processus.

Types des données échangés sur les canaux

Si les types des données échangées sur les canaux sont différents, ceux ci peuvent être pris en compte.

Fréquences des communications sur les canaux

Les deux méthodes d'évaluation des coûts de communication ci-dessus deviennent inadéquates dès que les processus contiennent des communications dans des boucles pour lesquelles on ne connait pas le nombre de pas d'itérations. Il en est de même lorsque le programme contient des communications dans les branches d'un choix.

Antonelli et Vannechi dans [Antonelli89] proposent u ne méthode empirique pour des programmes généraux écrits en ECSP et comportant des boucles. Cette méthode est basée sur une observation (que les auteurs disent largement admise) que tous les programmes passent une partie du temps dans les boucles. On utilise alors une évaluation du temps de communication comme dépendant du niveau d'imbrication de la boucle.
Sarkar dans [Sarkar89a] propose une méthode basée sur les fréquences d'exécution pour l'évaluation quantitative du volume des communications et des temps d'exécution. Les fréquences d'exécution sont obtenues par "profiling". L'auteur utilise des compteurs mis directement dans le programme. Ces compteurs sont incrementés à l'exécution et stockés dans une base de données associée au programme à la fin de chaque exécution.

1.3.1.2 Evaluation des coûts d'exécution

Le temps d'exécution d'une tâche est en général évalué en nombre d'instructions élémentaires. Comme pour les communications, le temps d'exécution d'une tâche peut dépendre du jeu de données en entrée et donc varier d'une exécution à l'autre.

Les méthodes d'évaluation du temps d'exécution les plus courantes utilisent des techniques de profiling qui collectent des données sur plusieurs exécutions et calculent un temps moyen d'exécution pour les tâches. Une telle méthode est décrite dans [Sarkar89a]. On peut aussi utiliser des techniques de profiling qui mesurent le temps réel d'exécution des tâches. Cette technique a été utilisée par Boillat et al. dans leurs outils de configuration automatique de programmes OCCAM, MARC et PFY [Boillat91].

1.3.2 L'allocation - l'ordonnancement

L'allocation consiste à affecter les diverses tâches aux divers processeurs dans le but de minimiser le temps d'exécution parallèle du programme. Le temps d'exécution parallèle dépend du taux d'utilisation des processeurs et de la surcharge induite par la communication entre les processeurs.

Le problème d'allocation d'un ensemble de tâches sans communications a été étudiée par divers auteurs et une présentation détaillée en est faite dans [Graham79].

On sait par exemple que l'allocation d'un ensemble de tâches sans communications et sous des contraintes de précédence a été montré NP-Complet si les tailles des tâches sont quelconques [Garey79, Sarkar89b].

Le problème d'allocation devient plus compliqué lorsqu'on prend en compte les communications. Alors que l'exploitation du parallélisme pousse à utiliser des processeurs différents chaque fois que des tâches peuvent être exécutées en parallèle, la charge induite par les communications pousse à affecter sur le même processeur les tâches qui communiquent beaucoup. Les deux objectifs étant contradictoires, l'allocation s'efforcera de trouver un bon compromis entre parallélisme et surcharge des communications [Talbi93].

1.3.3 Contrôle des communications et synchronisations

Le compromis entre parallélisme et communication fait qu'à la fin de l'étape d'allocation, des tâches qui communiquent peuvent être affectées sur des processeurs non directement connectés. D'autre part, les divers langages de programmation utilisés offrent un ensemble de modèles d'interaction (protocoles) qui impliquent une coordination entre tâches localisées sur des processeurs distants.

La coordination entre tâches distantes nécessite un contrôle de leurs communications et synchronisations. La mise en œuvre de ce contrôle se ramène à quelques mécanismes de base dont les plus importants sont :

- un mélange de synchronisation et de diffusion sur tout ou partie de l'ensemble des processeurs,
- une communication point à point efficace pour permettre l'accès distant à l'environnement mis en commun lors de l'interaction,
- des mécanismes efficaces d'équilibrage de charge permettant d'assurer que les participants d'une interaction arriveront à leur point de synchronisation commune au même moment.

Une bonne référence où l'on peut trouver les divers modèles d'interaction entre entités d'une application parallèle ainsi qu'un panorama des langages pour la programmation parallèle est l'article de Bal et al. [Bal89]. Deux modèles de base y sont discutés : l'échange de messages et les données partagées. Leur mise en œuvre sur une architecture parallèle y est aussi analysée.

L'éventail des modèles d'interaction conduit à la nécessité pour un système d'exploitation d'offrir un support pour la construction efficace de plusieurs protocoles; un protocole a deux fonctions principales : le "transport" d'un message entre l'émetteur et le récepteur, et le respect d'un certain mode de synchronisation. La construction effective du protocole sur une machine parallèle va donc se traduire par deux types de messages : ceux transportant les données effectives, et ceux de contrôle utilisés pour le respect du protocole.

Un noyau de communication d'un système d'exploitation pour machines parallèles doit donc assurer les deux fonctions suivantes :

- virtualisation du réseau d'interconnexion de la machine en proposant un service d'acheminement des messages entre les processeurs, et
- mise en œuvre effective des protocoles en générant au besoin les divers messages de contrôle.

La première fonction correspond à la communication entre processeurs et est fortement dépendante de l'architecture de la machine cible. La deuxième fonction est d'un niveau plus haut et correspond à la communication entre les processus d'une application.

Communication entre processus : protocoles

La communication entre processus repose sur le service d'acheminement et doit offrir une interface transparente indépendante de l'architecture. Elle nécessite les opérations suivantes :

- Une désignation des entités du système (processus et objets de communication) indépendante de la localisation. La mise en œuvre de ce service implique des communications systèmes.
- Le découpage et réassemblage des messages en paquets.
- Le respect de la sémantique des protocoles, en générant des messages de contrôle comme les requêtes d'établissement de communication et les accusés de réception.

Communication entre processeurs : routage des messages

Le routage consiste à offrir un service d'acheminement des messages entre n'importe quelle paire de processeurs. La virtualisation du réseau d'interconnexion en un réseau logique complètement connecté convient très bien à des protocoles qui ne nécessitent que des communications point à point. Lorsque le protocole implique plus de deux entités, la communication point à point pour envoyer un message à plusieurs destinataires introduit un encombrement du réseau par des messages transitant plusieurs fois sur les mêmes processeurs.
Pour optimiser l'utilisation des ressources, il est nécessaire pour le service d'acheminement de supporter, en même temps que la communication point à point, l'envoi d'un même message à plusieurs processeurs (diffusion).
Plus généralement, on pourrait s'intéresser à un acheminement offrant des services plus élaborés tels que ceux offerts dans le matériel pour la synchronisation, la combinaison de données, le "piggy-backing" (réponses aux messages) etc.

ParX : un noyau de système pour systèmes parallèles

ParX est un noyau de système parallèle développé au sein de notre équipe de recherche [Gonzalez91, Langue91, Balaniuk93, Castro93]. Le noyau de communication de Parx, comme le montre la figure 1.1, comporte deux niveaux. Le premier niveau, appelé couche de routage, est une extension du matériel et réalise la communication entre les processeurs. Deux types d'acheminement sont supportés dans ParX : la communication point à point et la diffusion d'un message à un groupe de processeurs. Les deux types d'acheminement de ParX cohabitent et sont basés sur une même fonction de routage générale qui garantit l'abscence d'interblocage.

Le second niveau assure l'interprétation des messages et la communication entre les processus suivant un protocole donné.

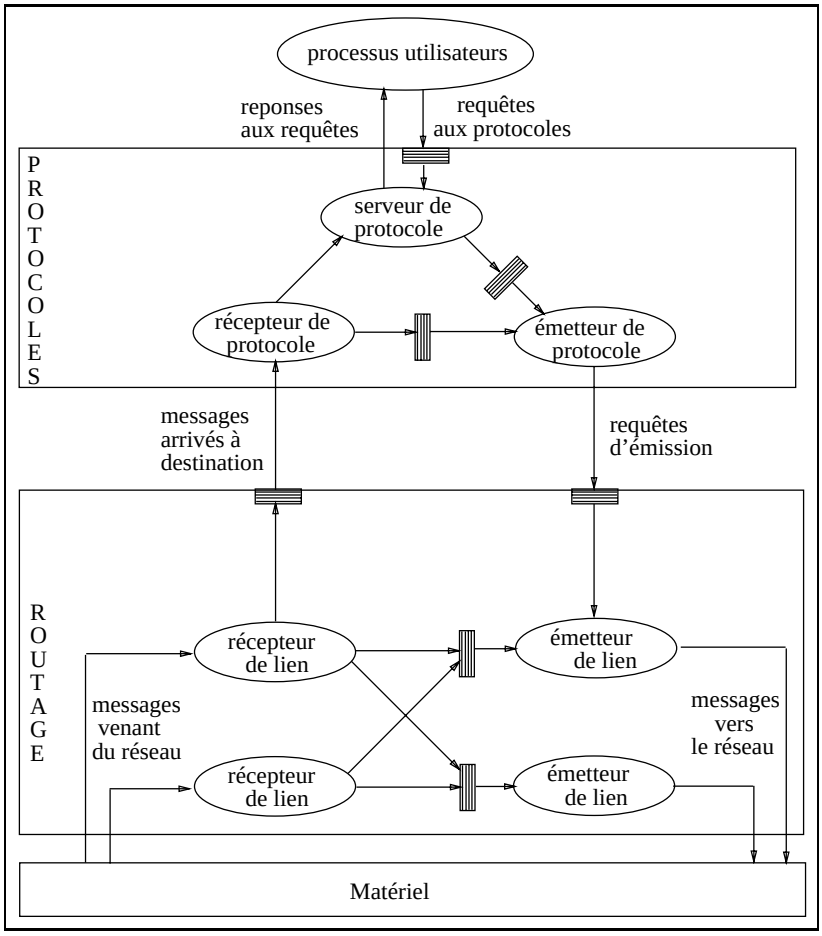

Figure 1.1: Noyau de communication de Parx

La réalisation du niveau d'interprétation des protocoles dans ParX est décrite dans [Gonzalez91] et [Langue91]. Le travail présenté dans ce rapport concerne la partie acheminement point à point, et plus particulièrement l'étude des fonctions

de routage qui est notre principale contribution dans ParX.

1.4 Contribution de cette thèse

L'objet de l'étude présentée ici est l'acheminement des messages dans les machines massivement parallèles. L'accent est mis sur des solutions extensibles utilisant un nombre de ressources indépendant de la taille de la machine.
Nous avons étudié le problème sur deux volets : le routage sur un réseau d'interconnexion fixe, et la communication par reconfiguration dynamique d'un réseau d'interconnexion programmable en reliant à la demande les processeurs voulant communiquer.

La communication par reconfiguration dynamique pose le problème du coût du réseau d'interconnexion. A travers l'exemple des machines hiérarchiques supernodes dont le réseau d'interconnexion est basé sur des réseaux de Clos 3-étages, nous montrons que la complexité du contrôle des réseaux d'interconnexion multi-étages est grande, et que donc la communication par reconfiguration dynamique est difficilement envisageable dans des machines de grande taille. La reconfiguration est alors utilisée par phases à l'intérieur desquelles on utilise le routage sur des réseaux d'interconnexion fixes dont la topologie peut être quelconque car celle-ci dépend uniquement des besoins des applications et des possibilités offertes par la machine reconfigurable considérée.

Le routage comporte essentiellement deux composantes : une technique d'acheminement des messages qui définit les traitements appliqués dans chaque nœud et la fonction de routage qui définit les chemins suivis par les messages. Quelle que soit la technique d'acheminement utilisée, on est amené à partager un ensemble de ressources (tampons de stockage ou liens physiques) et donc à des risques d'interblocage. Nous proposons une nouvelle méthode de génération de fonctions de routage sans interblocage pour réseaux généraux. Notre méthode, contrairement aux méthodes classiques qui procèdent par ordonnancement des tampons, nécessite un nombre de tampons par processeur indépendant de la taille de la machine.

La nouvelle génération des machines parallèles intègre de plus en plus de fonctions dans le matériel, notamment la fonction d'acheminement des messages. Pour que cette intégration soit la plus efficace possible, des méthodes nouvelles de représentation compacte de l'information de routage doivent être utilisées.
Nous nous sommes intéressés à une technique de représentation compacte de l'information de routage qui utilise, pour chaque processeur, une table de routage dont la taille est de l'ordre du nombre de ses voisins. Cette technique dite du routage par intervalles a été proposée en 1983 par Santoro et Khatib [Santoro82]. Elle consiste à étiqueter les processeurs du réseau et à associer un intervalle de

l'ensemble des étiquettes à chaque voisin d'un processeur ; l'intervalle associé à un voisin représentant l'ensemble des processeurs que l'on peut atteindre en passant par ce voisin.

Malgré le fait que le routage par intervalles ait été choisi par INMOS pour ses nouveaux processeurs de communication C104 [INMOS91], peu de méthodes de génération de tables de routage par intervalles existent. Notamment, très peu de travaux ont été menés sur les méthodes de génération de fonctions de routage qui sont sans interblocage. Dans un premier temps, nous considérons le cas d'un tore et proposons une nouvelle méthode de routage par intervalles. Tout en garantissant l'abscence d'interblocage, notre méthode conduit à des chemins dont la longueur est proche de l'optimum. Nous nous intéressons ensuite aux réseaux généraux, et là aussi nous proposons une méthode de génération de tables de routage par intervalles qui sont sans interblocage.

Enfin, la classe des fonctions de routage qui peuvent être décrites par des tables de routage par intervalles est réduite par rapport à celle des fonctions qui utilisent des tables de routage non compactes. Tout en gardant une information de routage par processeur dépendant uniquement du nombre de voisins, nous proposons une extension du routage par intervalle, le schéma d'étiquetage étendu (SEE). Le SEE permet, entre autres choses, de décrire des fonctions de routage adaptatives et contient le routage par intervalle classique. Une méthode de génération de SEE pour réseaux généraux est aussi proposée.

1.5 Organisation de la suite du rapport

La suite de ce rapport est divisé en deux parties.

La première, regroupant les chapitres 2, 3 et 4, vise à présenter les problèmes de l'acheminement de messages. Le chapitre 2 étudie à travers l'exemple des machines supernodes l'acheminement de messages par reconfiguration dynamique. Les chapitres 3 et 4 concernent l'acheminement de messages par routage. Au chapitre 3 nous faisons un état de l'art sur les techniques d'acheminement et les méthodes de prévention de l'interblocage. Notre méthode de génération de fonctions de routage sans interblocage est présentée au chapitre 4.

La deuxième partie traite du compactage de l'information de routage et regroupe les chapitres 5, 6 et 7. Au chapitre 5 nous posons le problème du compactage de l'information de routage et faisons un état de l'art sur la technique de routage par intervalles. Le chapitre 6 présente de nouvelles méthodes de génération de fonctions de routage par intervalles, et au chapitre 7 nous présentons notre nouvelle technique de représentation compacte de l'information de routage, le SEE.

Partie I

Un mécanisme de base pour la communication : l'acheminement des messages

Chapitre 2

Acheminement des messages par reconfiguration dynamique

2.1 Introduction

Les machines reconfigurables ont pour objectif de faciliter une communication efficace entre processeurs non voisins grâce à un dispositif de commutation des liens. Celui-ci permet en effet de réaliser par programme divers types de réseaux d'interconnexion.

La réalisation d'un réseau d'interconnexion sur une machine reconfigurable comprend deux phases :

- Le choix du placement des sommets du graphe du réseau d'interconnexion sur les processeurs physiques,
- Le contrôle de la configuration qui consiste à déterminer les commandes de connexion à exécuter par le dispositif de commutation.

Les dispositifs de commutation des machines parallèles sont inspirés de ceux des réseaux de téléphones ; à ce titre ce sont des réseaux de permutation composés de points de connexion élémentaires dont les performances attendues sont :

1. un faible coût matériel,
2. un faible délai de traversée,
3. la réarrangeabilité : réalisation de n'importe quelle permutation,
4. la simplicité du contrôle

En général les propriétés 3 et 4 sont couvertes par des réseaux dits *non bloquants* dont la caractéristique est de pouvoir établir une nouvelle connexion indépendamment de celles qui existent.

Le coût matériel d'un réseau de permutation est souvent évalué par le nombre de points de connexions élémentaires (crosspoints). Le réseau de permutation le plus simple est un commutateur matriciel à N entrées et M sorties (encore appelé crossbar $N \times M$). Le crossbar vérifie les propriétés 2, 3, et 4 ; par contre la propriété 1 dépend du nombre de processeurs à interconnecter. En effet un crossbar $N \times M$ connectant N entrées à M sorties nécessite NM points de connexion élémentaires. Ce coût est trop élevé pour les réseaux de grande taille. A titre d'exemple la technologie actuelle ne permet de construire que des crossbars de l'ordre de $200X200$, ce qui permet, en considérant des processeurs à 4 liens bidirectionnels, de construire de machines avec une quarantaine de processeurs.

L'optimisation du nombre de points de connexion élémentaires d'un crossbar pour des réseaux de grande taille se fait souvent au détriment de son caractère non bloquant et donc de son contrôle ; on se contente alors de la propriété de *réarrangeabilité*. Des études d'optimisation, menées notamment par Clos [Clos53], ont abouti entre autres à des dispositifs à base de crossbars dits réseaux multi-étages au nombre desquels on compte les réseaux de Clos, de Benes, les "Baselines", "shuffle exchange ", "data manipulator", etc.. On trouvera une étude détaillée de tous ces réseaux dans [Feng81].

Parmi les machines reconfigurables, nombreuses sont celles dont le dispositif de commutation est un simple crossbar [Waille90, Bhatkar90]. Pour construire de grosses machines on utilise des réseaux multi-étages [Waille90, Beeten87].

Dans ce chapitre nous nous intéressons à l'acheminement des messages dans les machines reconfigurables dont le dispositif de commutation est un réseau multi-étages. Nous prenons comme exemple les machines hiérarchiques supernodes (supernodes dans la suite) issues du projet européen ESPRIT P1085 auquel notre équipe a participé dès sa proposition en 1985, et aujourd'hui construites par les sociétés Telmat et Parsys. Le dispositif de commutation des supernodes est basé sur le principe des réseaux de Clos réarrangeables à 3 étages.

Ce chapitre est organisé de la façon suivante : dans la section 2.2, après quelques rappels sur les réseaux de Clos à 3 étages, nous donnons une description schématique et succincte de l'architecture des supernodes ; une description plus complète est présentée dans [Waille90]. Dans la section 2.3, nous modélisons le problème du contrôle des réseaux de Clos en termes de coloriage des arêtes d'un graphe biparti régulier, puis présentons et analysons les algorithmes de contrôle.

2.2 Architectures des supernodes

2.2.1 Réseaux de Clos à 3 étages

Un réseau de Clos à 3 étages $C(n, m, r)$ connectant $N = nr$ entrées à N sorties est composé de r crossbars $n \times m$ au premier étage (blocs d'entrées), m crossbars $r \times r$ au deuxième étage (blocs intermédiaires), r crossbars $m \times n$ au troisième étage (blocs de sorties). Les étages sont interconnectés selon la règle suivante illustrée par la figure 2.1.

- la sortie j du bloc d'entrées i est reliée à l'entrée i du bloc intermédiaire j.
- la sortie k du bloc intermédiaire p est reliée à l'entrée p du bloc de sorties k.

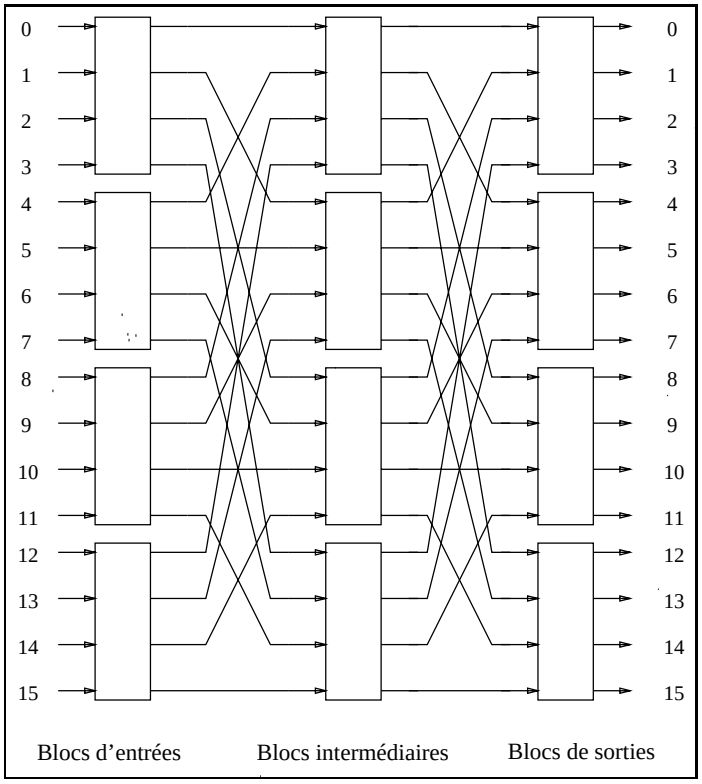

Figure 2.1: Réseau de Clos $C(4, 4, 4)$

Deux résultats fondamentaux sur les réseaux de Clos sont :

Théorème 2.1 (Clos) *[Clos53] : Une condition nécessaire et suffisante pour qu'un réseau de Clos $C(n, m, r)$ soit non bloquant est : $m \geq 2n - 1$.*

Théorème 2.2 (Slépian-Duguid) *[Paull62] : Une condition nécessaire et suffisante pour qu'un réseau de Clos $C(n, m, r)$ soit réarrangeable est : $m \geq n$.*

On trouvera dans [Clos53] une étude détaillée des coûts des réseaux de Clos multi-étages en fonction des caractéristiques de fonctionnement souhaitées. On y trouve notamment les plages des valeurs des paramètres pour lesquelles le réseau de Clos, en dehors du délai de traversée, devient supérieur au crossbar.

2.2.2 Dispositif de commutation des supernodes

Les supernodes sont une famille de machines parallèles reconfigurables à base de transputers comprenant :

- 16 à 1024 processeurs
- des processeurs de service,
- des processeurs de contrôle,
- Un dispositif de commutation permettant de connecter les différents processeurs entre eux,
- Un bus servant à l'échange d'informations de contrôle (signaux de contrôle, de synchronisation etc.) entre les processeurs de contrôle, de travail et de service.

L'organisation générale (générique) de toutes ces composantes est donnée par la figure 2.2 pour une machine supernode à 256 processeurs.

Le dispositif de commutation des supernodes comprend quatre réseaux de permutation identiques et indépendants (réseaux de base). Chacun de ces quatre réseaux de base réalise l'un des quatre types de connexion suivants : $Nord$ vers Sud (N/S), Sud vers $Nord$ (S/N), Est vers $Ouest$ (E/O) et $Ouest$ vers Est (O/E) où $Nord$, Sud, Est et $Ouest$ sont des étiquettes des 4 liens d'un transputer (figure 2.3). Dans un supernode à N transputers, chacun des quatre réseaux de base connecte donc N entrées à N sorties.

La configuration du supernode consiste à réaliser un réseau d'interconnexion donné à l'aide des quatre réseaux de base. Elle nécessite donc une étape préliminaire d'étiquetage des arêtes du graphe du réseau d'interconnexion comme Nord-Sud ou Est-Ouest. Cet étiquetage, qui détermine les connexions devant être faites dans chacun des quatres réseaux de base, pourra être réalisé par la méthode de partage par cycle eulérien décrite dans [Nicole88]. Les résultats de cet étiquetage sont deux permutations Π_1 et Π_2 sur les sommets du graphe du réseau d'interconnexion correspondant respectivement aux connexions étiquetées N/S et E/O. Chaque transputer étant connecté à la même entrée sur chacun des quatre réseaux de base, et les connexions à réaliser étant bidirectionnelles, les réseaux N/S (S/N) réalisent Π_1 $({\Pi_1}^{-1})$ et les réseaux E/O (O/E) Π_2 $({\Pi_2}^{-1})$.

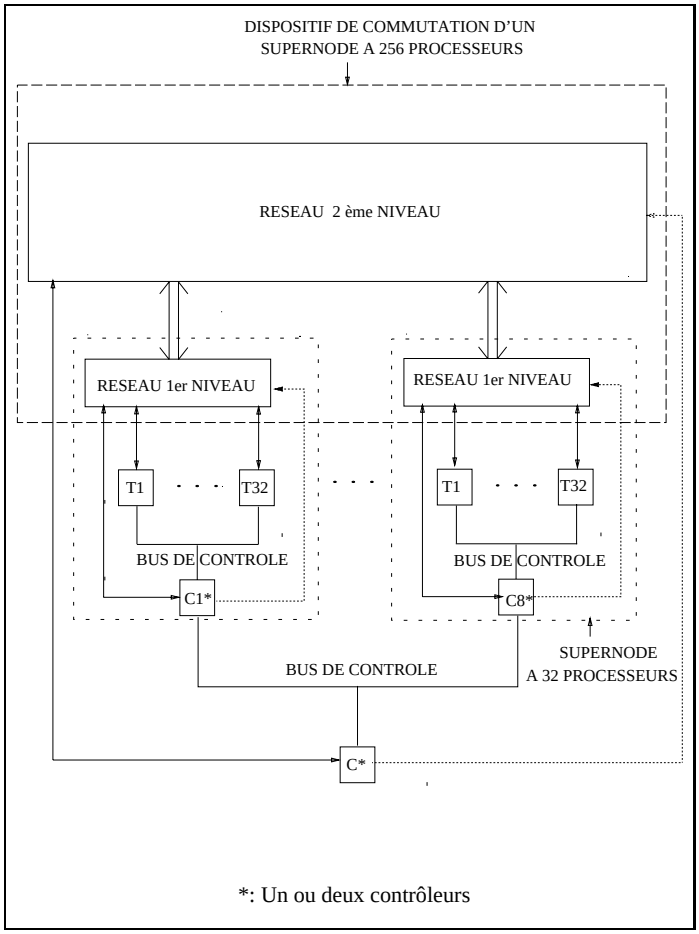

Figure 2.2: Architecture d'un supernode à 256 processeurs.

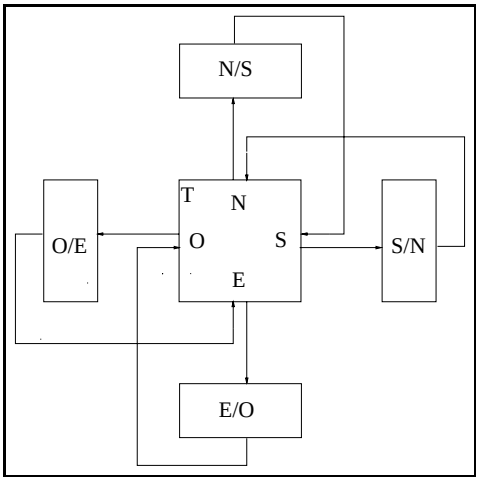

Figure 2.3: Connexion d'un transputer aux 4 réseaux de base

Le réseau de base d'un supernode comportant $N = nr$ transputers est un réseau de Clos "3 étages - 2 niveaux" composé de r crossbars $2n \times 2n$ au premier niveau (jouant le rôle des 1^{er} et 3^{me} étages) et de m crossbars $r \times r$ au 2^{me} niveau (jouant le rôle de l'étage intermédiaire). Remarquons que la fusion des blocs d'entrées et de sorties $n \times n$ en des blocs $2n \times 2n$ du 1^{er} niveau permet de connecter les processeurs appartenant à un même bloc en ne traverssant qu'un crossbar au lieu de 3 pour un réseau de Clos.

2.3 Algorithmes de configuration des supernodes

La configuration d'un supernode comportant $N = nr$ transputers se ramène à résoudre le problème de la configuration d'un réseau de base sachant que celui-ci contient un réseau de Clos $C(n, n, r)$. En vertu du théorème de Slépian-Duguid, celle-ci est toujours possible quelle que soit la permutation à réaliser. Par contre, parce que les paramètres du réseau de base ne remplissent pas les conditions du théorème de Clos, il est des configurations dont les connexions ne peuvent être établies indépendamment les unes des autres.
Dans cette section nous considérons le réseau de Clos contenu dans un réseau de base et examinons les algorithmes pour son contrôle. Nous supposons que les connexions à réaliser impliquent toutes les entrées et sorties du réseau. Nous commençons par présenter une modélisation du problème, les algorithmes sont ensuite décrits et analysés. Avant de présenter le modèle, rappelons quelques notions de la théorie des graphes.

2.3.1 Définitions

Pour les propriétés relatives aux concepts définis, le lecteur pourra consulter [Berge83]. Soit $G = (V, E)$ un graphe non orienté, où V est l'ensemble des sommets et E celui des arêtes.

- Un *couplage* de G est un ensemble d'arêtes non adjacentes 2 à 2
- Un couplage est dit *maximum* si et seulement si son cardinal est maximum.
- Un *couplage* est dit *maximal* s'il n'est contenu dans aucun autre.
- Un sommet v est dit *saturé* par un couplage M s'il existe une arête de M incident à v.
- Un couplage est dit *parfait* s'il sature tous les sommets.
- G est dit simple s'il admet au plus une arête entre deux sommets.
- On appelle *chaîne alternée* par rapport à un couplage M, une chaîne simple (n'utilisant pas 2 fois la même arête) dont les arêtes sont alternativement dans M et $E - M$.

- Nous appelons *extenseur* par rapport à un couplage M, une chaîne alternée par rapport à M et allant d'un sommet insaturé à un autre sommet insaturé.

2.3.2 Modélisation du contrôle

Soit Π la permutation de $\{0, \ldots, N-1\}$, où $N = nr$, correspondant à une configuration d'un réseau de Clos $C(n, n, r)$. Associons à Π le graphe biparti $G_\Pi = (X, Y, E)$ où $X = \{x_0, \ldots, x_{r-1}\}$ est l'ensemble des blocs d' entrées, $Y = \{y_0, \ldots, y_{r-1}\}$ est l'ensemble des blocs de sorties, et une arête de E représente une connexion entre un bloc d'entrées et un bloc de sorties. Il y a autant d'arcs entre un bloc d'entrées et un bloc de sorties qu'il y a d'entrées de l'un à relier aux sorties de l'autre. Chaque arc peut être étiqueté par la connexion qu'il représente. G_Π est régulier de degré n. La figure 2.4 montre le graphe G_Π associé à la réalisation sur un réseau de Clos $C(4, 4, 4)$ d'une configuration définie par la permutation $\Pi = (13, 7, 3, 12, 8, 9, 5, 10, 14, 1, 0, 4, 6, 2, 15, 11)$ de $\{0,\ldots,15\}$.

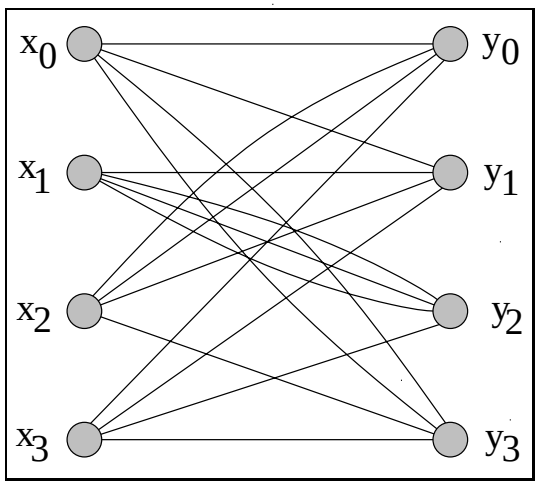

Figure 2.4: Graphe biparti G_Π.

La réalisation de la permutation Π consiste à affecter chaque arête de G_Π à un bloc intermédiaire par lequel la connexion sera réalisée. Sous la contrainte matérielle du réseau de Clos que deux connexions à un même bloc (d'entrées ou de sorties) ne peuvent être affectées au même bloc intermédiaire, on retrouve le problème du coloriage des arêtes du graphe biparti G_Π en autant de couleurs qu'il y a de blocs intermédiaires, ou ce qui est équivalent, le problème de l'extraction de n couplages parfaits deux à deux disjoints. Les figures 2.5 et 2.6 indiquent un coloriage du graphe G_Π de l'exemple précédent et la configuration induite sur le réseau de Clos.

Une autre modélisation consiste à considérer la matrice d'interconnexion de G_Π : $H_\Pi = (H_\Pi(i, j))$ $0 \leq i, j \leq r-1$ où $H_\Pi(i, j)$ est le nombre de connexions entre les blocs x_i et y_j et à en étudier la décomposition en une somme de n ma-

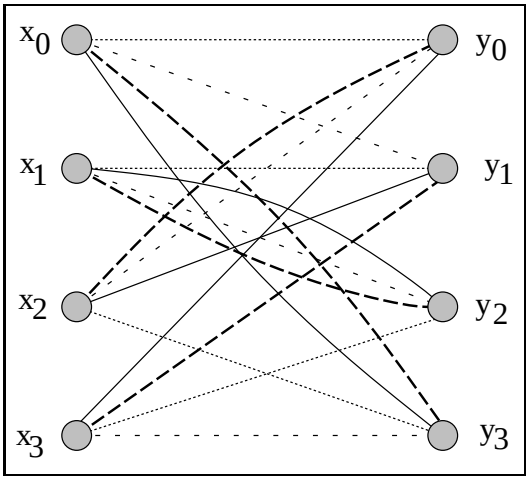

Figure 2.5: Graphe biparti G_Π colorié.

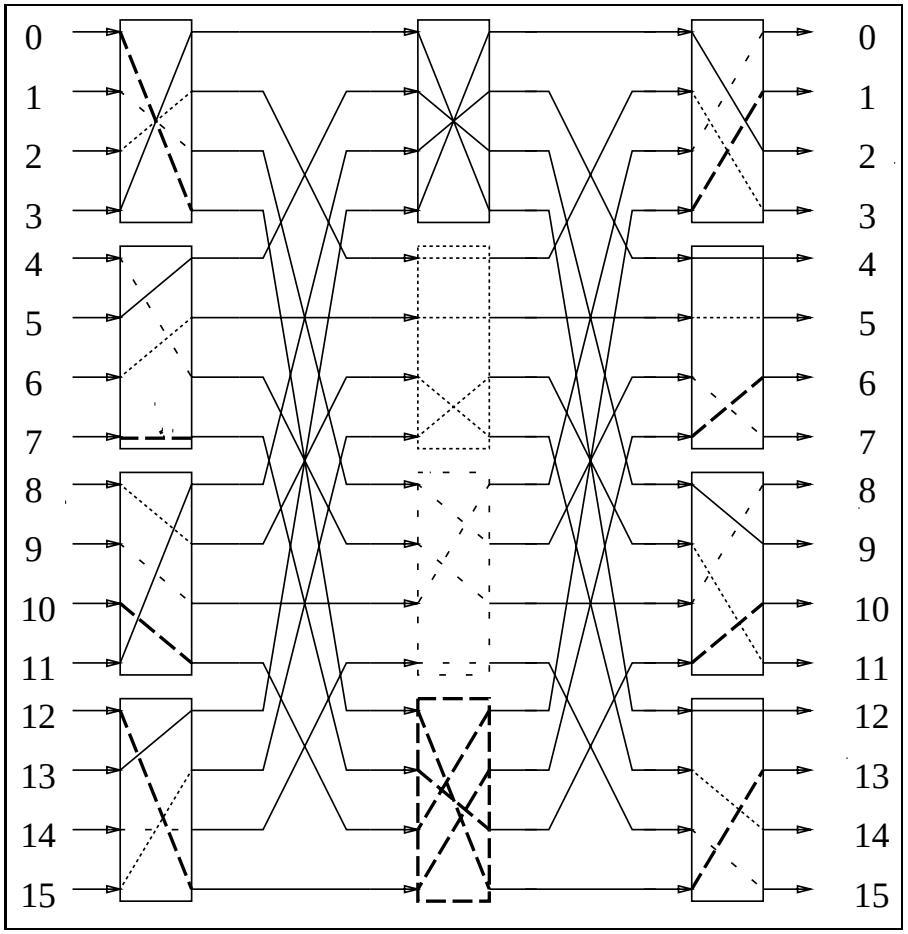

Figure 2.6: Configuration du réseau de Clos $C(4,4,4)$ pour la permutation Π

trices de permutation (une matrice de permutation correspondant à un couplage) [Neiman69, Tsaowu74]. Pour l'exemple précédent on a :

$$H_{\Pi} = \begin{bmatrix} 1 & 1 & 0 & 2 \\ 0 & 1 & 3 & 0 \\ 2 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Les matrices de permutation associées au coloriage de la figure 2.5 sont respectivement

$$\begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \quad \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad et \quad \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

pour le 1^{er}, 2^{me}, 3^{me} et 4^{me} blocs du 2^{me} étage.
Notons enfin qu'une variante de l'approche matrice d'interconnexion a été utilisée par Ramanujam dans [Ramanujan73] sous l'appellation "allocator matrix". Il s'agit de la matrice d'interconnexion enrichie des étiquettes des connexions : l'élément $H_{\Pi}(i,j)$ est cette fois-ci l'ensemble des arêtes reliant x_i à y_j.
Tous ces formalismes conduisent aux mêmes algorithmes que nous présentons ci-dessous en termes d'extraction de couplages parfaits dans un graphe biparti régulier de degré n.

2.3.3 Algorithmes de configuration

A la base de tous les algorithmes de calcul d'un couplage maximum des sommets d'un graphe se trouve le théorème de Berge qui s'énonce comme suit :

Théorème 2.3 (Berge) *[Berge83] : Une condition nécessaire et suffisante pour qu'un couplage M soit maximum est qu'il n'existe pas d'extenseur par rapport à M.*

Ce théorème conduit à un algorithme itératif d'extraction d'un couplage maximum. En effet, soit M un couplage non maximum. D'après le théorème de Berge, G contient un extenseur par rapport à M, appelons le C. On peut alors construire un couplage M' tel que $|M'| = |M| + 1$. Il suffit de prendre $M' = M \bigoplus C$, c'est à dire de substituer dans C toutes les arêtes de $E - M$ à celles de M et inversement[1]. Sur cette itération, appelée dans la suite itération de Berge, est basée une procédure d'extraction d'un couplage parfait qu'on peut écrire comme suit :

[1] $M' = (M - C) \bigcup (C \bigcap (E - M))$

Procédure Couplage_Parfait(X,Y,E)
début

on affecte les arêtes à M de façon gloutonne */
1. M := couplage initial maximal

2. Tant Que $|M| \neq r$ faire
2.1 $C := Extenseur(X, Y, E, M)$
2.2 $M := M \oplus C$
Fin Tant que
RETURN(M)
fin

L'étape étiquetée 1 consiste à calculer un premier couplage maximal, en affectant les arêtes au couplage M d'une façon gloutonne (sans remise en cause des choix déjà faits). On montre (voir [Tsaowu74]) qu'à la fin de cette étape $|M| \geq r/2$ quelque soit la technique gloutonne utilisée. Le corps de la boucle 2 (lignes étiquetées 2.1 et 2.2) n'est rien d'autre que l'itération de Berge décrite plus haut.

La recherche d'un extenseur par rapport à un couplage M s'effectue en construisant tous les arbres dont la racine, sommet de niveau 0, est insaturée et dont les arêtes reliant les sommets de niveau pair (impair) à ceux de niveau impair (pair) sont dans $E - M$ (M). En d'autres termes, dans ces arbres, les chemins menant de la racine aux feuilles sont des chaînes alternées par rapport à M. M étant non maximum, le théorème de Berge nous dit qu'un des arbres ainsi construits contient un sommet feuille insaturé. Mais, le graphe G considéré étant biparti régulier, on montre aisément que quel que soit le sommet insaturé choisi comme racine son arbre contient une feuille insaturée. Il suffit donc de construire un seul arbre, et nous avons l'algorithme suivant pour la recherche d'un extenseur. Pour en simplifier la description, nous supposons que le graphe (X,Y,E) est simple.

Procédure Extenseur(X,Y,E,M)
/*
- *marque* est un tableau de booléens indexé sur Y initialisé à $faux$
$marque[y] = vrai$ ssi le sommet y a été visité
- *prcdent* est un tableau indexé sur $X \bigcup Y$ servant à représenter
les chaînes alternées.
- S est une file, l'opération d'entrée en file sera l'union ($\bigcup$)
- La fonction $premier(S)$ donne le premier élément de S et le supprime de S
*/
début
x := un sommet de X insaturé par M
$S := \{t \in Y/(x,t) \in E \; et \; marque[t] = faux\}$
$prcdent[x] := NIL$

$y := premier(S)$
Tant que y saturé par M faire
$marque[y] := vrai$
$prcdent[y] := x$
$x :=$ le sommet $s \in X/(s,y) \in M$
$prcdent[x] := y$
$S := S \bigcup \{t \in Y$ tel que $(x,t) \in (E-M)$ *et* $marque[t] = faux\}$
$y := premier(S)$
fin Tanque
$C := \emptyset$
Tant que $prcdent[y] \neq NIL$ faire
$C := C \bigcup \{(y, prcdent[y])\}$
$y := prcdent[y]$
fin Tanque
RETURN(C)
fin

Les procédure d'extraction de couplage parfait et de recherche d'extenseur sont à la base des algorithmes de contrôle des réseaux de Clos que nous présentons ci-dessous et dont nous évaluons la complexité.

2.3.3.1 Algorithme de Neiman

L'algorithme de Neiman [Neiman69, Tsaowu74] consiste à extraire les couplages parfaits un à un. On a la procédure suivante :

Procédure Neiman
/*
- $G = (X, Y, E)$ est le graphe biparti régulier de degré n
- La procédure Neiman extrait n couplages parfaits de G
*/
début
Pour $i = 1$ à n faire
$M_i :=$ *Couplage_Parfait(X,Y,E)*
$E := E - M_i$
Fin Pour
fin

Complexité

L'algorithme de Neiman comporte une boucle externe Pour s'exécutant n fois et dont corps consiste en un appel à la procédure *Couplage_Parfait*. Dans cette procédure, l'étape étiquetée 1 a une complexité en $O(r^2)$ et l'étape étiquetée 2

a pour composante principale la recherche d'un extenseur dont la complexité est en $O(r^2)$. La boucle Tant que s'exécutant au plus $r/2$ fois, le corps de la boucle Pour a un coût total en $O(r^3)$. Ce qui donne un coût en $O(n\,r^3)$ pour l'algorithme de Neiman.

2.3.3.2 Algorithme de Vizing

Contrairement à l'algorithme de Neiman, l'algorithme de Vizing [Gabow76] mène tous les couplages de front. Aussi, après avoir construit de façon gloutonne n couplages maximaux, se pose le problème de leur réorganisation pour augmenter leur cardinalité. Soit donc $e = (x, y)$ une arête non affectée. Soit deux couplages M_i saturant x et ne saturant pas y et M_j saturant y et ne saturant pas x. De tels couplages existent sinon e aurait été affecté à un couplage. Pour affecter e, par exemple à M_i, il suffit, sur la plus longue chaîne C d'origine x alternant des arêtes de M_i et de M_j, de réaffecter à M_i (resp. M_j) les arêtes de M_j (resp. M_i). La procédure est la suivante :

Procédure Vizing
/*
- T est l'ensemble des arêtes non encore affectées à un couplage
- La fonction $premier(T)$ donne le premier élément de T et le supprime de T
*/
début
1. Pour $i = 1$ à n faire
   /* Affecter les arêtes à M_i de façon gloutonne */
   $M_i :=$ Couplage maximal initial
   Fin Pour
   $T := E - \bigcup_{1 \le i \le m} M_i$
2. Tant que $|T| > 0$ faire
   $e := premier(T)$ /* $e = (x, y)$ */
2.1 $i :=$ numéro du couplage saturant x et ne saturant pas y
2.2 $j :=$ numéro du couplage saturant y et ne saturant pas x
   $C :=$ la plus longue chaîne partant de x et construite en
   alternant les arêtes de M_i et celles de M_j.
2.3 $M_j := M_j \bigoplus C$
2.4 $M_i := (M_i \bigoplus C) \bigcup \{e\}$
   $T := T - \{e\}$
   Fin tant que
Fin

Complexité

L'étape 1 consiste à affecter les arêtes aux couplages sans remettre en cause les décisions déjà prises. On montre (voir [Tsaowu74]) qu'à la fin de cette étape

chaque couplage M_i contient au moins $r/2$ arêtes. A la fin de l'étape 1 au moins $nr/2$ arêtes ont donc été affectées à des couplages. La recherche d'un couplage initial maximal a un coût en $O(r^2)$. L'étape 1 de l'algorithme de Vizing a donc un coût en $O(nr^2)$. L'étape 2 consiste en une boucle Tant que s'exécutant au plus $nr/2$ fois. Seules les lignes étiquetées 2.1 à 2.4 ont un coût non constant. 2.1 et 2.2 ont un coût en $O(n)$. Le coût de 2.3 et 2.4 est la longueur de la chaine C et est donc en $O(r)$. L'algorithme de Vizing a donc un coût en $O(\max(nr^2, n^2r))$.

2.3.3.3 Algorithmes basés sur le principe du "divide and conquer"

Il est bien connu que tout graphe biparti G dont les sommets sont de degré pair n peut être partitionné en deux sous graphes bipartis G' et $G"$ dont les sommets sont de degré $n/2$. Il suffit pour cela de procéder à une partition par cycle eulérien c'est à dire, pour chaque composante connexe de G, parcourir un des cycles eulériens associés en affectant alternativement ses arêtes à G' et $G"$ [Nicole88]. Donc si $G_\Pi = (X, Y, E)$ est un graphe biparti régulier dont les sommets sont de degré pair n alors $G' = (X, Y, E')$ et $G" = (X, Y, E")$ sont bipartis réguliers de degré $n/2$. Cette propriété est exploitée par les algorithmes basés sur le principe du "divide and conquer". On distingue 2 cas :

Cas $n = 2^k$ avec $k > 0$

On génère au bout de k partitions par cycle eulérien des graphes bipartis ($G_i = (X, Y, E_i)$; $1 \leq i \leq n$) dont les sommets sont de degré 1, donc des couplages parfaits de G_Π. Cet algorithme est en $O(nr \log_2 n)$. Connue également sous le nom de "looping algorithm" [Andresen77], cette technique a été utilisée par Waksman dans [Waksman68] pour une décomposition optimale, en crossbars 2×2, d'un réseau de permutation .

Cas général : Algorithme de Gabow

Dans le cas où n n'est pas une puissance de 2, la technique précédente génère des graphes intermédiaires G_i dont le degré est impair et on ne peut donc pas procéder à une partition par cycle eulérien. Gabow dans [Gabow76] propose alors d'appliquer un algorithme d'extraction de couplage parfait pour se ramener à un graphe de degré pair. L'algorithme de Gabow s'écrit comme suit :

Procédure Gabow(G)
/*
- i est une variable globale initialisée à 0 avant l'appel initial
- $G = (V, E)$
- La procédure *partition_Euler* appliquée à un graphe G de degré pair, retourne deux Graphe G' et $G"$ resultats d'une partition de G par cycle eulérien. */

début
  $i := i + 1$
  si $n = 1$ alors $M_i := E$
  sinon
    si n impair alors
1. $M_i :=$ Couplage_Parfait(G)
      $E := E - M_i$
    fin si
2. (G', G") = partition_Euler(G)
3. Gabow(G')
4. Gabow(G")
  fin si
fin

Complexité

Dans le pire des cas, on a des graphes de degré impair après chaque partition par cycle eulérien. Ce cas se présente quand $n = 2^k - 1$. Soit U_i le coût de l'algorithme de Gabow appliqué à un graphe de degré $2^i - 1$. On a $U_i = r^3 + (2^i - 2)r + 2U_i - 1$ avec $U_1 = 0$. La résolution de cette équation récurrente donne un coût en $O(\max(r^3 n, nr \log_2 n))$.

2.4 Acheminement de messages par reconfiguration

Les algorithmes présentés dans ce chapitre, même celui basé sur la technique du "divide and conquer", ont en commun la remise en cause d'une décision d'affectation d'une arête de G à tel ou tel couplage dont les conséquences dépendent de la nature des applications. Dans une application au comportement déterministe, dont toutes les configurations sont connues, celles-ci peuvent être calculées à l'avance et on pourra s'y référer comme bibliothèque. Ainsi il n'y a pas de dégradation de performance des applications du fait d'un calcul des commandes de configuration à l'exécution.

Toutefois, la synthèse d'un réseau d'interconnexion répondant exactement aux besoins d'une application donnée n'est pas un problème facile [Talbi93], et en général on devra être ammené à changer les connexions dans le réseau en cours de fonctionnement. La reconfiguration comme mécanisme d'acheminement peut alors s'avérer fort préjudiciable car elle peut impliquer l'interruption d'un ensemble de communications en cours. Le reste de ce rapport est consacré au routage.

Chapitre 3

Acheminement des messages par routage

3.1 Introduction

Dans le chapitre précédent nous avons montré sur un exemple les limites de l'acheminement des messages par reconfiguration dynamique. Nous nous intéressons ici au routage.
Le routage comporte essentiellement deux composantes :

- une technique d'acheminement des messages qui définit les traitements appliqués dans chaque nœud, et
- une fonction de routage qui détermine les chemins suivis par les messages.

Après une analyse des problèmes posés par la réalisation du routage et un état de l'art sur les techniques d'acheminement communément utilisées, nous nous intéressons à la fonction de routage. Nous en déterminons les caractéristiques essentielles à la correction et aux performances du noyau de communication d'une machine massivement parallèle.

3.2 Organisation et fonctionnement d'un routeur

La solution la plus simple pour l'organisation du routeur consiste à considérer un groupe de tampons qui servent pour stocker tout message transitant sur le processeur avant sa retransmission. Le schéma d'un tel routeur est donné par la figure 3.1. Chaque lien d'entrée a un processus récepteur dont le programme consiste à recevoir un message, l'étiqueter avec un lien de sortie fourni par la fonction de routage, et le déposer dans un tampon. Le processus associé au lien de sortie va chercher dans le groupe de tampons les messages qui lui sont destinés et les retransmet sur le lien.

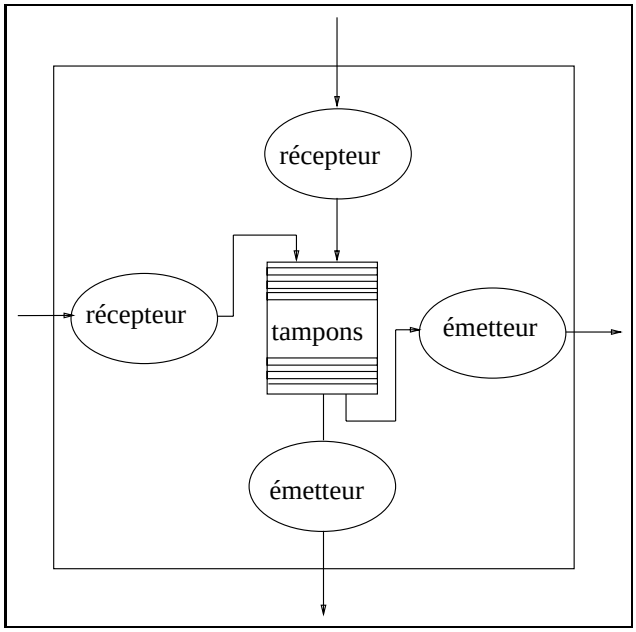

Figure 3.1: Organisation d'un routeur avec une seule classe de tampons

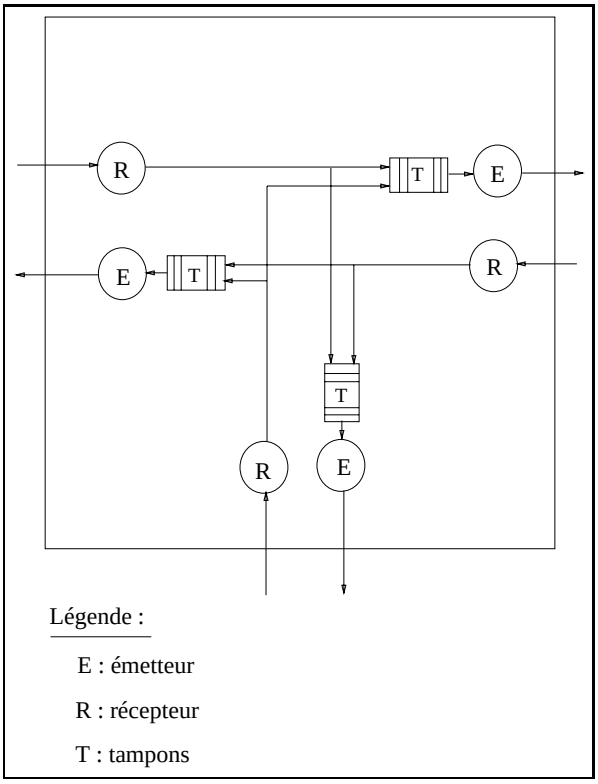

Figure 3.2: Organisation d'un routeur avec une classe de tampons par lien

Cette organisation du routeur couple fortement les liens alors qu'en général ils peuvent fonctionner indépendamment. L'organisation la plus couramment utilisée consiste à utiliser une classe de tampons par lien de sortie (figure 3.2). Le processus de réception sur un lien dépose cette fois-ci le message directement dans les tampons associés au lien de sortie donné par la fonction de routage.
Dans la suite, sauf pour des raisons de simplification de la présentation, nous considérerons un routeur avec une classe de tampons par lien.

3.3 Techniques d'acheminement

Le routage des messages n'a pas ses origines dans les machines parallèles qui nous intéressent ici, il apparaît déjà vers les années 70 dans les premiers réseaux d'ordinateurs[Quillan77, Sunshine77, Ahuja79a, Ahuja79b, Quillan80]. Depuis cette époque, les techniques d'acheminement des messages ont fait l'objet de nombreuses recherches. Les solutions proposées peuvent être classées en deux catégories : la commutation des circuits, et la commutation des messages.

3.3.1 La commutation de circuits

La commutation des circuits est basée sur le principe de la téléphonie. Elle consiste en une réservation complète du chemin entre deux processeurs avant que l'échange ne commence. Un signal de réservation est envoyé par le processeur source vers le processeur destination. Passant d'un processeur à un autre, le signal réserve le lien traversé. Si à un instant donné il ne trouve aucun lien libre, suivant la stratégie de réservation utilisée, il en attend un ou signale au processeur source que la réservation est impossible. La transmission effective du message ne commence que quand le signal de réservation a atteint le processeur destination et que ce dernier renvoie un acquittement à la source. Le chemin n'est libéré qu'après réception complète du message.

On distingue la commutation des circuits logiques et la commutation de circuits physiques. La commutation de circuits physiques implique que le processeur a un moyen matériel de connecter directement un lien d'entrée à un lien de sortie pour que les messages puissent traverser le nœud sans être mémorisés.

Quoique la commutation de circuits puisse se révéler efficace pour l'acheminement de très longs messages, le taux d'utilisation des liens reste faible notamment pour la commutation de circuits logiques. C'est pour résoudre ce problème de la faible utilisation des liens que la commutation des messages a été proposée.

3.3.2 La commutation de paquets et de messages

3.3.2.1 La technique "store-and-forward"

La commutation de messages consiste à acheminer les messages pas à pas d'un processeur à un autre. A chaque processeur le message est mémorisé et retransmis vers le processeur suivant. Cette technique, connue dans la littérature sous le nom de "store and forward", présente l'inconvénient que si le prochain lien est libre, le message doit être entièrement reçu avant d'être retransmis.

Pour résoudre ce problème, la commutation de paquets implique simplement que le message à acheminer est découpé en plusieurs morceaux (paquets) de même taille. Ceci permet d'améliorer le temps de transmission dans le cas de très longs messages en profitant de l'effet pipeline. D'autre part, chaque paquet pouvant être acheminé indépendamment des autres, plusieurs paquets d'un même message peuvent être envoyés sur des liens différents, profitant ainsi de l'existence de plusieurs chemins entre deux nœuds.

Le découpage des messages en paquets implique un assemblage lorsque les messages sont arrivés à destination. L'assemblage des messages est simple dans le cas où les paquets entre deux nœuds donnés suivent le même chemin et ne se doublent pas. Dans le cas où l'une des deux conditions n'est pas verifiée l'assemblage des messages nécessite la numérotation des paquets.

Le délai de transmission d'un paquet est $\tau_{sf} = (L/B)D$ où L est la longueur du paquet, B la bande passante d'un lien et D la longueur du chemin suivi par le paquet.

3.3.2.2 Le "virtual cut-through", le "wormhole", et le "double stockage "

Même si le découpage des messages en paquets permet une bonne utilisation des liens de communication, le problème des liens libres non utilisés reste toujours posé dans le cas d'un paquet, ici encore il faut attendre la réception complète du paquet avant de le retransmettre.

Trois variantes ont été proposées dans la littérature pour résoudre ce problème : les techniques dites du "virtual cut-through" [Kermani79], du "wormhole " [Dally87], et du "double stockage " [Whobrey88]. Elles sont toutes basées sur le principe que si le prochain lien est libre on retransmet le paquet dès réception de l'entête. Si le réseau est peu chargé, la probabilité de trouver un lien libre est grande et on se rapproche de la commutation de circuits. Par contre, si la charge du réseau est grande, le paquet va être bloqué plusieurs fois et on se rapproche du store-and-forward.

Le délai de transmission est cette fois-ci $\tau = (L_h/B) \times D + L/B)$ où L_h est la longueur de l'entête et L la longueur du paquet.

Les trois variantes diffèrent dans le cas où le lien de sortie n'est pas libre. Alors que la technique "virtual cut-through " stocke le paquet dans le nœud où il est bloqué, le routage "wormhole " arrête la progression du paquet et le stocke donc dans plusieurs nœuds, et la technique du "double stockage " stocke le paquet à la source et tente de le réémettre plus tard.

Les techniques wormhole et virtual cut-through sont équivalentes lorsque le réseau est peu chargé. Lorsque le réseau devient chargé la technique virtual cut-through devrait être meilleure. En effet, un paquet dont la progression est stoppée ne peut bloquer que les paquets qui traversent le nœud où il y a eu blocage, alors que pour la technique wormhole d'autres paquets dans des nœuds intermédiaires peuvent être bloqués. Par contre, la technique wormhole a l'avantage de ne nécessiter que peu de tampons de stockage par nœud et est donc plus facile à implanter dans le matériel. La technique du double stockage présente le grand inconvénient que rien ne garantit qu'un paquet ne sera pas toujours recalé.

3.3.2.3 La technique dite du "mad-postman"

La technique du "mad-postman" [Jesshope89a, Jesshope89b, Miller91, Jesshope91], vise à supprimer le délai de stockage de l'entête du paquet. Elle a été jusqu'à présent utilisée par dans les grilles multi-dimensionnelles. Elle fonctionne de la façon suivante :

- Le réseau d'interconnexion comporte un ensemble de directions de routage.
- Sans attendre la réception complète de l'entête, le paquet est retransmis sur le lien de sortie dans la même direction que celle d'où il vient.
- Dès que l'entête est complètement reçue, la décision de routage est prise. Si elle est différente de celle anticipée, le paquet est émis dans la nouvelle direction. Si la décision était bonne, l'entête a déjà été transmise sans délai au niveau du nœud.

L'avantage de cette technique est que si l'anticipation sur la direction du paquet était bonne, le paquet traverse le nœud pratiquement sans délai. Le délai de transmission est alors L/B où L est la longueur du paquet et B la bande passante d'un lien. Dans le pire des cas, c'est à dire si le paquet change de direction à chaque pas, le délai de transmission est le même que pour les techniques wormhole et virtual cut-through.

L'inconvénient majeure de la technique du mad-postman est la génération d'entêtes errantes dans le réseau. Une procédure pour les reconnaître et les éliminer doit être mise en œuvre. Dans la réalisation décrite dans [Miller91] pour les grilles non rebouclées, les entêtes errantes peuvent être facilement identifiées soit quand elles

sont bloquées lorsque le lien à emprunter est occupé, soit quand elles arrivent au bord de la grille.

3.3.3 Sur la mise en œuvre des techniques

La technique du store-and-forward a été la plus utilisée car ne nécessitant pas un support matériel. Les autres techniques, quoiqu'elles puissent être simulées par logiciel, ne sont vraiment efficaces que si l'on dispose d'un support matériel permettant de connecter directement un lien d'entrée à un lien de sortie sans mémorisation.

La technique store-and-forward est notamment à la base des routeurs logiciels des premières générations de machines parallèles comme le cosmic-cube (prototype de recherche du caltech) et toutes les versions commerciales qui en découlent (IPSC, Ncube1), et réseaux de transputers T400 et T800. Les machines plus récentes disposent d'un support matériel. Ainsi, l'Intel IPSC-2 et IPSC/860 utilisent la commutation de circuits, le Ncube, le paragon, la J-machine et le processeur de routage C104 de INMOS utilisent le wormhole.

La technique du mad-postman a été implantée dans un processeur de communication pour la grille [Miller91, Jesshope91].
Peu d'implantations de la technique virtual cut-through ont été réalisées, citons tout de même le processeur de communication conçu dans le projet Haartz [Dolter89]. Une évaluation des performances de ce processeur sur un réseau en grille avec des connexions diagonales (hexagonal mesh) est présentée dans [Dolter91].

3.4 Les principaux problèmes du routage

La mise en œuvre de l'une ou l'autre des techniques d'acheminement implique le partage des ressources physiques du réseau d'interconnexion, à savoir les tampons de stockage et les liens physiques. On doit donc assurer un ensemble de propriétés essentielles à la correction et à un bon fonctionnement d'un système distribué au nombre desquelles l'absence d'interblocage, l'absence de famine, l'équité de service, la tolérance aux pannes, l'équilibre de charge, l'absence de bouclages infinis, etc.

3.4.1 Abscence de famine et équité

La famine et l'équité sont liées. On dit qu'il y a situation de famine lorsqu'un paquet ne peut accéder à un tampon alors que celui-ci devient de temps en temps libre mais est toujours pris par d'autres paquets en compétition.

Garantir l'équité consiste à traiter tous les paquets de la même façon sans qu'il y en ait qui soient privilégiés. Dans le cas général, l'équité entre tous les paquets

transitant sur le réseau n'est pas nécessaire, plusieurs niveaux de priorité sont utilisés et on cherche à assurer l'équité à un niveau de priorité donné et l'absence de famine entre les divers niveaux de priorité.

Les problèmes de famine et d'équité de service sont assez facilement résolus par le service d'acheminement en utilisant des stratégies de service telles que "premier arrivé - premier servi " pour une priorité donnée et le partage à la proportionnelle entre les priorités.

3.4.2 interblocage

On dit qu'il y a interblocage lorsqu'un ensemble de paquets est bloqué et ne peut avancer que si un paquet de l'ensemble libère un tampon. Gunther dans [Gunther81] distingue au moins 6 types d'interblocage dans un système de communication : direct, indirect, dû à l'assemblage des paquets, dû à l'attente d'accusé de réception (piggy-back deadlock), dû à des priorités pour certains paquets, et causé par les processus utilisateurs. Seuls les interblocages direct et indirect concernent le service de routage, les autres sont en général résolus par les niveaux supérieurs. Nous supposerons notamment que ces niveaux supérieures (protocoles) consomment les paquets arrivés à destination au bout d'un temps fini.

L'interblocage direct ne met en jeu que deux processeurs. Les tampons de l'un sont pleins de paquets dont la prochaine étape est l'autre processeur. Cet interblocage se résout facilement en utilisant deux groupes de tampons dans chaque nœud et en réservant un groupe pour les paquets en entrée et l'autre pour les paquets en sortie. L'interblocage direct ne peut donc pas se produire pour le type de routeur que nous considérons.

L'interblocage indirect peut survenir même si les précautions pour la prévention de l'interblocage direct ont été prises. Il concerne au moins trois processeurs. Il consiste en une attente cyclique entre N processeurs ; le processeur i attendant que son voisin $(i+1)$ mod N libère un tampon pour pouvoir lui envoyer un paquet. Ce type de situation est illustré par la figure 3.3 où les processeurs impliqués dans l'interblocage forment un anneau de 5 processeurs. Dans la suite, lorsque nous parlerons d'interblocage, c'est seulement à l'interblocage indirect que nous ferons référence.

Interblocage et graphe de dépendance des liens

L'interblocage est dû à la fonction de routage car c'est elle qui détermine les chemins suivis par les paquets. Il y a deux conditions nécessaires et suffisantes à l'interblocage :

1. La présence d'un cycle dans la demande d'utilisation des tampons, et

2. Tous les tampons sur le cycle sont pleins.

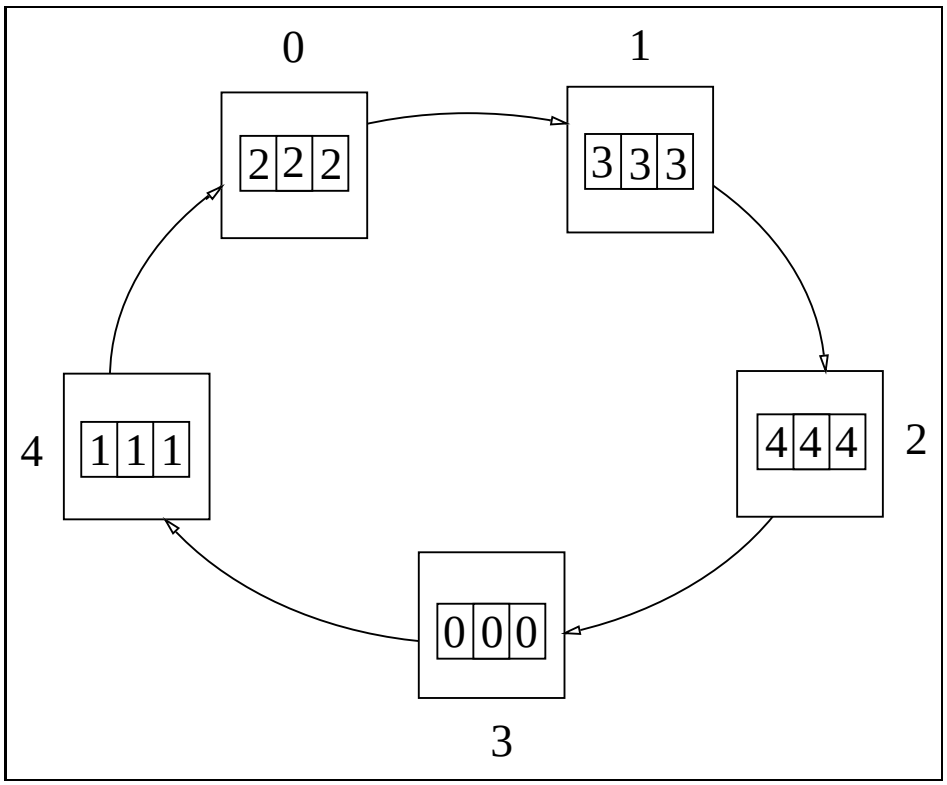

Figure 3.3: Un exemple d'interblocage indirect : chaque tampon est plein de messages destinés à son voisin à distance 2.

Les méthodes de lutte contre l'interblocage s'attachent à assurer que l'une ou l'autre des deux conditions ne soit pas verifiée. Dans le cas d'un routeur où à chaque lien de sortie est associé un groupe de tampons banalisés, Dally et Seitz [Dally87] ont introduit la notion de graphe de dépendances associé à une fonction de routage pour modéliser la demande d'utilisation des tampons.

Définition 3.1 (Graphe de dépendances) *Soit G le graphe d'un réseau d'interconnexion. Le graphe de dépendances associé à une fonction de routage R est un graphe orienté dont les sommets sont les liens (unidirectionnels) de G dans lequel il y a un arc (l_1, l_2) entre de l_1 vers l_2 si et seulement si l_1 est un prédécesseur direct de l_2 dans un chemin induit par R.*

Le graphe de dépendances permet donc d'exprimer la première condition nécessaire à l'interblocage. Nous y aurons souvent recours pour montrer que les solutions proposées sont sans interblocage.

3.4.3 Validité

Contrairement à l'interblocage où un ensemble de paquets sont bloqués et ne peuvent plus progresser, il peut arriver qu'un paquet tourne indéfiniment dans

le réseau sans pouvoir atteindre la destination. Garantir la propriété de validité consiste à assurer qu'en l'absence d'interblocage tous les paquets arriveront à destination.

Comme l'interblocage la propriété de validité est liée à la fonction de routage. Des fonctions de routages non valides peuvent être utilisées si chaque processeur n'a pas besoin de communiquer avec tous les autres. Dans la suite nous ne faisons aucune hypothèse sur les besoins de communication, nous nous intéressons donc uniquement à des fonctions de routage valides.

3.5 La fonction de routage

3.5.1 Les modes de routage

Il y a principalement deux modes de routage, déterministe et adaptatif. Le routage déterministe n'utilise qu'un seul chemin (calculé a priori), entre chaque paire de processeurs.

Le routage adaptatif, lui, tient compte des variations du trafic dans le réseau. Le chemin suivi par un paquet est construit au fur et à mesure que celui-ci progresse en fonction du trafic réel. Le routage adaptatif nécessite donc le maintien et la mise à jour de l'état de charge des liens.

Un routage adaptatif peut être pur, dans ce cas il n'y a pas de restrictions sur les liens de retransmission d'un paquet. Il peut aussi choisir le prochain lien sur le chemin dans un ensemble de liens calculé à l'avance, on parle alors de routage semi-adaptatif.

3.5.2 Représentation de la fonction de routage

La fonction de routage détermine les chemins que doivent suivre les paquets. A chaque couple (source, destination), elle associe donc l'ensemble de tous les chemins entre la source et la destination. Cette façon de représenter la fonction de routage est la plus générale mais nécessite beaucoup de place. En général on se ramène à une fonction qui, au lieu de l'ensemble de tous les chemins, donne l'ensemble de tous les successeurs possibles. La représentation d'un chemin est donc répartie sur le réseau. On a donc une fonction R qui, en chaque nœud intermédiaire (appelé dans la suite processeur courant) et pour chaque destination, fournit les liens par lesquels le paquet peut sortir pour atteindre la destination.

$$R(processeur\ courant, destination) = lien\ de\ sortie$$

Dans sa mise en œuvre effective, la fonction de routage peut utiliser d'autres paramètres comme le processeur source, le lien d'entrée du paquet et la charge

des liens. Ceci pour atteindre divers objectifs comme l'absence d'interblocage et l'équilibrage de la charge sur le réseau. Par exemple, la méthode de génération de fonctions de routage sans interblocage que nous proposons au chapitre 4 utilise, en plus du processeur courant et de la destination, le lien par lequel le paquet est entré.

3.6 Méthodes de résolution du problème d'interblocage

L'utilisation de la concurrence dans les ordinateurs s'est accompagnée de nombreuses recherches sur la résolution du problème d'interblocage. Les principales situations d'interblocage dans la gestion des ressources des systèmes informatiques sont présentées dans [Isloor80]. Deux écoles de pensée prévalent ; l'une procède par détection-guérison [Chan87, Cidon87], et l'autre par prévention [Gopal85, Gunther81]. Nous présentons brièvement la technique de détection-guérison ; la prévention de l'interblocage est étudiée dans la section suivante.

3.6.1 Techniques de détection-guérison

Les techniques de détection-guérison de l'interblocage fonctionnent de la façon suivante :

- Aucune mesure n'est prise "a priori " pour éviter l'interblocage.
- Un algorithme de détection des cas d'interblocage est mis en œuvre en même temps que les procédures normales de routage.
- Lorsque un interblocage est détecté, un mécanisme de guérison est activé pour l'éliminer.

Une telle méthode est décrite dans [Chan87]. Ces techniques ne sont pas adaptées pour les machines massivement parallèles pour les principales raisons suivantes :

- Elles requièrent l'échange de paquets de contrôle contenant des listes de liens pouvant être impliqués dans un interblocage. La taille de ces listes peut être grande et être de l'ordre du nombre de processeurs dans le réseau.
- Les algorithmes de détection utilisés sont complexes car ils se ramènent toujours à la recherche de cycles dans le réseau. Cette recherche devient coûteuse dans le cas des réseaux de grande taille.
- La détection est basée sur une observation du comportement des files d'attente. Des délais de garde sont souvent utilisés pour vérifier l'activité des files. La détermination d'un délai de garde qui convient à toutes les situations n'est pas simple, car devant prendre en compte l'activité réelle du réseau.

De par la surcharge qu'elles introduisent à l'exécution, les méthodes de détection-guérison de l'interblocage ne sont pas adaptées aux machines parallèles. Dans la suite nous nous intéresserons uniquement à la prévention de l'interblocage.

3.7 Techniques de prévention de l'interblocage

Les techniques de prévention de l'interblocage assurent que l'une au moins des deux conditions nécessaires à l'interblocage ne peut pas être remplie. Elle sont toutes basées sur des restrictions dans l'utilisation des tampons. Les techniques de prévention de l'interblocage peuvent être classées en deux catégories suivant qu'elles nécessitent un contrôle dynamique de l'admission des paquets dans les tampons ou que l'utilisation des tampons est entièrement déterminée avant l'exécution.

3.7.1 Méthode nécessitant un contrôle à l'exécution

Ces méthodes visent surtout à assurer que la deuxième condition nécessaire à l'interblocage ne soit pas remplie. Une discipline régit l'admission des paquets dans la couche de routage et assure qu'il y a au moins un tampon libre sur tout ensemble de processeurs pouvant être impliqué dans une situation d'interblocage. Trois techniques utilisent ce principe : la méthode de Gelernter, la méthode de Roscoe, et une méthode proposée par Blazewics et al. basée sur l'estampillage.

3.7.1.1 Méthode de Gelernter

Soit G le graphe bidirectionnel du réseau d'interconnexion. La méthode de Gelernter [Gelernter81] consiste à orienter G de façon à ce qu'il n'y ait pas de cycle et que tous les processeurs puissent être atteignables à partir d'un processeur source r. L'orientation consiste à construire un graphe (orienté) G_1 comme suit:

- Partant de r, on construit un arbre recouvrant en orientant les liens de la racine vers les feuilles
- Un lien qui n'appartient pas à l'arbre est orienté du nœud le plus près de la racine vers le nœud le plus éloigné
- Un lien entre deux nœuds à égale distance de la racine est orienté dans un sens ou dans l'autre en évitant de former un cycle

La figure 3.4 donne un exemple de graphe G_1 sur une grille 3×3 dont les 4 coins sont reliés en anneau, la racine étant le nœud étiqueté r.
Lors du routage, un paquet est classé de type G_1 s'il doit emprunter un lien dans le sens donné par G_1, de type G_2 dans le cas contraire. Si le paquet a le choix entre plusieurs liens de sortie (routage adaptatif) dont l'un au moins est à contre sens de G_1, il est classé de type G_2.

L'absence d'interblocage est garantie en imposant que chaque nœud contienne à tout moment un tampon vide ou un paquet de type G_2. Cette contrainte est assurée par les deux règles de routage suivantes :

- Si, après avoir accepté un nouveau paquet, un nœud non racine trouve que tous ses tampons contiennent des paquets de type G_1, un paquet doit être choisi et rerouté sur un lien de type G_2.

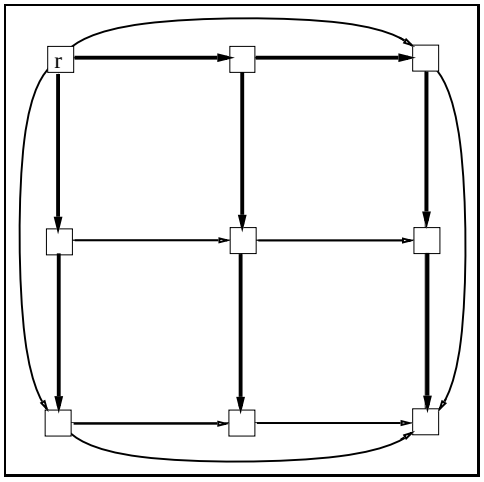

Figure 3.4: un graphe acyclique orienté G_1

- La racine r ne pouvant contenir que des paquets de type G_1, doit toujours avoir un tampon de libre même si c'est au prix de la perte d'un paquet

La méthode de Gelernter n'utilise que 2 classes de tampons par nœud. Par contre elle a deux inconvénients : la perte potentielle des paquets à la racine, et le reroutage des paquets. De par le fait qu'elle nécessite un reroutage des paquets, elle ne peut pas être utilisée dans le cadre d'un routage déterministe où les chemins sont calculés statiquement.

3.7.1.2 Méthode de Roscoe

La méthode de Roscoe [Roscoe87] est basée sur une discipline d'admission des paquets dans le service d'acheminement. Dans le cas d'un anneau unidirectionnel, chaque nœud est muni de deux tampons et l'absence d'interblocage est assuré en imposant la contrainte qu'un nœud n'accepte un paquet venant d'un utilisateur que si les deux tampons sont libres.

Dans le cas d'un réseau général, l'auteur propose de reconstruire la structure cyclique par l'utilisation, soit d'un circuit hamilitonien [1], soit en construisant un circuit hamilitonien virtuel sur un arbre recouvrant dans lequel on introduit des nœuds virtuels. La figure 3.5 illustre la construction d'un tel circuit sur une grille 4-4. Chaque nœud virtuel correspond à une classe de tampons et donc, dans le cas général, chaque nœud physique est muni de 2 classes de tampons contenant chacune au moins deux tampons.
L'absence d'interblocage est assurée par les règles suivantes :

- Un nœud n'accepte un paquet venant d'un utilisateur que si les deux tampons sont libres

[1] Dans le cas où le réseau considéré en admet un.

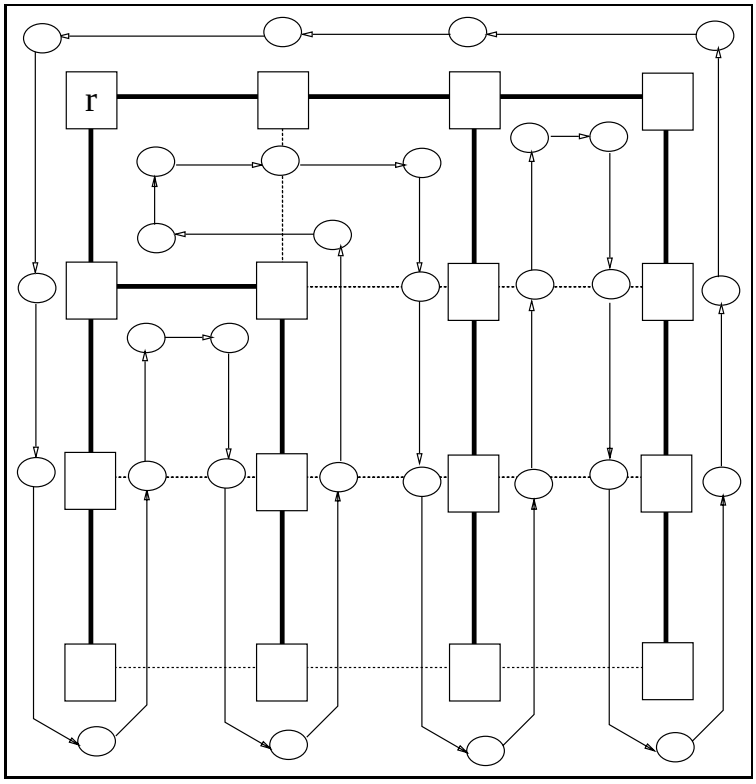

Figure 3.5: Circuit hamilitonien sur un arbre

- Les liens qui n'appartiennent pas au circuit peuvent être utilisés à condition que les paquets transmis dessus soient considérés comme ceux venant des utilisateurs. Un paquet émis sur un lien qui n'est pas dans le circuit peut donc être refusé. Il est alors réémis sur le circuit.

On doit donc attendre un accusé de réception pour les paquets envoyés sur les liens qui n'appartiennent pas au cycle. La technique de Roscoe est essentiellement dynamique et ne peut donc pas être utilisée dans le cadre d'un routage déterministe.

3.7.1.3 Methode basée sur l'estampillage

La méthode basée sur l'estampillage proposée par Blazewicz et al. [Blazewicz87] associe aux paquets une date d'arrivée dans le réseau. L'absence d'interblocage est assurée en imposant la contrainte qu'un paquet ne doit pas être bloqué par des paquets plus récents. Pour cela, à chaque fois qu'un paquet est bloqué par des paquets plus récents, un paquet est choisi et est échangé avec le "vieux" paquet en utilisant des tampons spéciaux réservés à cet effet.
Cette technique nécessite donc au moins un tampon pour l'acheminement dans chaque nœud, et au moins un tampon pour l'échange à chaque fois qu'on a deux processeurs connectés.

Le mécanisme d'échange implique un reroutage du paquet le plus récent qui peut être éloigné de sa destination. Notons toutefois que le paquet arrivera forcément à destination puisque dans le pire des cas il arrivera un moment où il sera le plus "vieux" dans le réseau.

Comme dans les deux autres techniques, le contrôle à l'exécution peut amener à dévier un paquet de sa route ; la méthode de Blazewicz ne peut donc pas être utilisée dans le cas du routage déterministe.

3.7.2 Méthodes basées sur les graphes de tampons

Les méthodes basées sur des graphes acycliques de tampons sont les plus utilisées. Elles consistent à répartir les tampons d'un nœud en plusieurs classes. On construit alors un graphe orienté de classes de tampons et les paquets ne sont transmis d'un tampon vers un autre que s'il y a un arc de la classe du premier vers celle du second. Le graphe des tampons doit avoir les propriétés suivantes :

- Pour chaque route dans le réseau, il y a au moins un chemin dans le graphe de tampons.
- Le graphe de tampons ne contient aucun cycle.

Bien que le nom ait été utilisé pour la première fois par Merlin et Schweitzer dans [Merlin80], cette méthode regroupe un ensemble de techniques qui ne diffèrent que par la technique de construction du graphe de tampons. La plus ancienne, le "structured buffer pool", a été proposée par Gunther dans le cadre du projet GMD-net [Gunther75]. Nous décrivons ci-dessous les diverses méthodes de construction du graphe de tampons.

3.7.2.1 Comptage de sauts (hops so far)

Dans la méthode dite du "comptage de sauts" [Giessler78], les paquets arrivant sur un processeur sont répartis par classes en fonction, par exemple, de la distance parcourue depuis le processeur source. On construit alors un graphe orienté sans cycle sur l'ensemble des classes. Pour garantir l'absence d'interblocage, chaque fois qu'un paquet traverse un lien physique sa classe est incrémentée de 1.
La méthode de comptage de sauts nécessite donc un nombre de classes de tampons égal à la longueur du plus long chemin plus 1. Si le routage est effectué sur les chemins les plus courts, au moins $D+1$ classes de tampons sont nécessaires dans chaque nœud, où D est le diamètre du réseau. Les figures 3.6 et 3.7 illustrent, pour les deux types de routeur, la construction du graphe de tampons par la méthode de comptage de sauts pour un réseau où la longueur maximum des chemins est 2.
Une variante de cette méthode décrite dans [Merlin80] consiste à considérer la distance qui reste à parcourir au lieu de la distance déja parcourue.

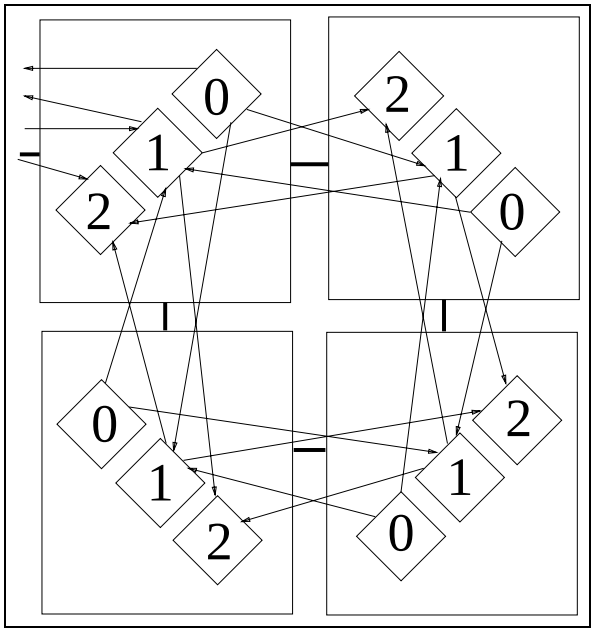

Figure 3.6: Graphe de tampons : méthode de comptage de sauts. Cas du routeur avec tampons communs aux liens de sortie.

3.7.2.2 Comptage de vallées

La technique dite de "comptage de vallées" a été proposée par Gunther et est décrite dans [Gunther81, Merlin80]. Les nœuds du réseau reçoivent un numéro différent chacun. Un chemin dans le réseau peut être représenté par la suite des numéros des nœuds traversés. Un nœud est dit *vallée* sur un chemin donné, s'il est suivi et précédé par des nœuds de numéros plus grands. Un nœud est dit *pic* s'il est précédé et suivi par des nœuds de numéros plus petits.
Soit K le plus grand nombre de vallées sur un chemin. Le graphe de tampons est construit en associant $2K+2$ classes de tampons ($u_0, u_1, \ldots, u_K, d_0, d_1, \ldots, d_K$) à chaque nœud. Les tampons de la classe u_i (d_i) seront occupés par les paquets qui "montent" vers un pic ("descendent" vers une vallée). Etant donnés deux nœuds n_s et n_t ($n_s < n_t$) entre lesquels il existe un lien, on a les règles de connexion suivantes pour la construction du graphe de tampons :

- Dans chaque nœud, la classe u_i $(0 \leq i \leq K)$ est connecté à la classe d_i sur le même nœud
- Dans chaque nœud, d_i $(0 \leq i < K-1)$ est connecté à u_{i+1} sur le même nœud
- La classe u_i $(0 \leq i \leq K)$ de n_s est connecté à la classe u_i de n_t
- La classe d_i $(0 \leq i \leq K)$ de n_t est connecté à la classe d_i de n_s

Le routage se fait donc de la façon suivante :

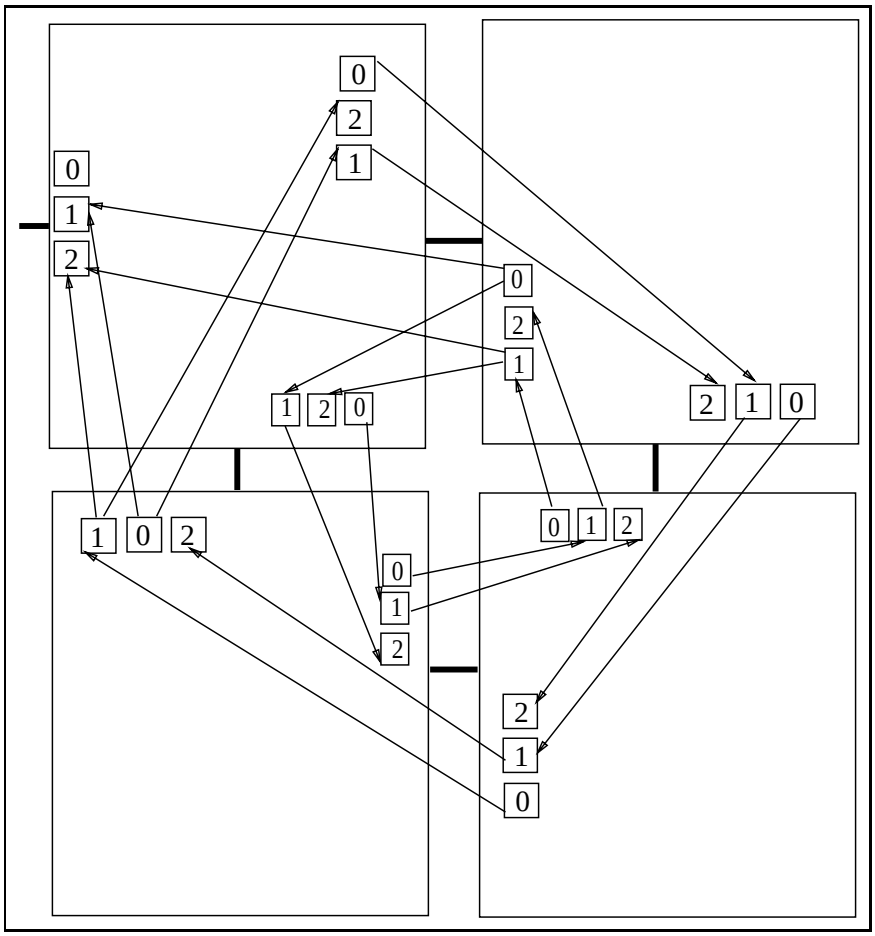

Figure 3.7: Graphe de tampons : méthode de comptage de sauts. Cas du routeur avec tampons différents pour les liens de sortie.

- Le paquet est introduit dans u_0.
- Les pas d'un nœud non vallée vers un nœud de numéro supérieur se font dans les tampons de classe u_i.
- Les pas d'un nœud non pic vers un nœud de numéro inférieur se font dans les tampons de classe d_i.
- Un paquet qui quitte un pic passe d'un tampon de classe u_i vers le un tampon dans la classe d_i du nœud suivant.
- Un paquet qui quitte une vallée passe de la classe d_i vers la classe u_i du nœud suivant.

Le nombre de classes de tampons requises par cette méthode est de $2(K+1)$, où K est le plus grand nombre de vallées qu'on peut trouver sur un chemin. La minimisation de K repose sur une méthode d'étiquetage des nœuds du réseau qui minimise le nombre de vallées sur un chemin.
Enfin, cette méthode nécessite que l'ensemble des chemins soit connu à l'avance. Elle ne peut donc être utilisée que dans un mode de routage déterministe ou un mode semi-adaptatif où le choix est fait dans un ensemble de chemins fixés à l'avance.

3.7.2.3 Comptage de sauts négatifs

La méthode dite de "comptage de sauts négatifs" ("negative hop scheme ") proposée par Gopal dans [Gopal85] consiste à partionner l'ensemble des nœuds du réseau de telle sorte que deux nœuds adjacents ne soient pas dans la même partition. Ceci revient au problème classique de coloriage des nœuds d'un graphe. Une telle partition est illustrée par la figure 3.8 où trois couleurs (0, 1, et 2) sont nécessaires pour une grille 3-3 dont les coins sont en plus connectés en anneau.

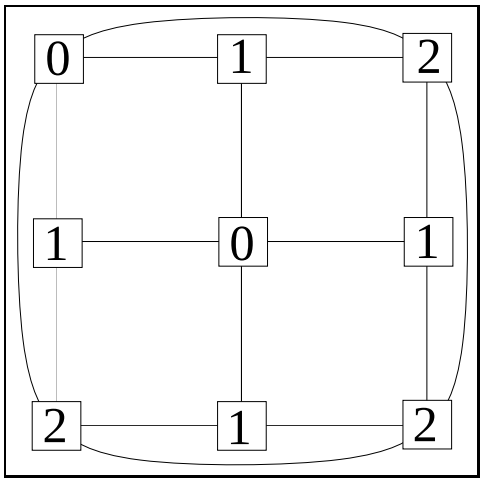

Figure 3.8: Coloriage d'une grille 3-3 (3 couleurs sont nécessaires)

On dit qu'un saut est négatif s'il permet de passer d'un nœud appartenant à une partion i vers un nœud d'une partion j avec $j < i$. Dans le cas contraire le saut est dit positif.
Soit M le nombre maximun de sauts négatifs que l'on peut trouver sur un chemin. Le graphe des tampons est construit de la façon suivante :

- Chaque nœud est muni de $M + 1$ classes de tampons.
- Entre deux nœuds A et B connectés, la classe i dans A a un arc vers la classe $i + 1$ dans B si le passage de A vers B est un saut négatif, et un arc vers la classe i dans le cas contraire.

Les paquets sont introduits dans la classe 0 ; dans un nœud donné la classe i est donc réservé aux paquets qui ont déjà effectué i sauts négatifs.

La minimisation du nombre de classes de tampons consiste à minimiser le nombre de sauts négatifs sur un chemin. Le nombre de sauts négatifs peut être borné en fonction du nombre P de partitions et de la longueur du plus long chemin D. Dans [Gopal85] on montre que $M < ((P-1)/P)D$. Une façon de minimiser le nombre de classes de tampons consiste donc à minimiser P. Pour un réseau quelconque, la détermination du nombre minimum P de partitions est un problème NP-complet (problème de coloration d'un graphe [Garey79]). L'auteur propose une amélioration de la méthode qui consiste à attribuer les étiquettes aux nœuds de telle sorte que tous les sous-graphes induits par les nœuds ayant la même étiquette soient sans cycle.

Remarquons que dans cette technique un protocole d'échange doit être utilisé à l'exécution pour prévenir l'interblocage direct dans le cas d'un routeur où les liens de sorties partagent les mêmes tampons.
La figure 3.9 montre un étiquetage de la même grille que dans la figure 3.8. Cette fois-ci deux étiquettes suffisent. Malheureusement dans le cas général on ne sait pas borner le nombre de couleurs.

3.7.2.4 Méthode dite du "Class climbing"

La méthode du "Class climbing" proposée par Annot et al. [Annot87] consiste à construire K graphes orientés acycliques. Le graphe des tampons est construit de la façon suivante :

- Chaque nœud est muni de K classes de tampons (une classe de tampons par graphe acyclique)
- Entre deux nœuds connectés il y a un arc entre les classes i et un arc de la classe i vers la classe $i + 1$

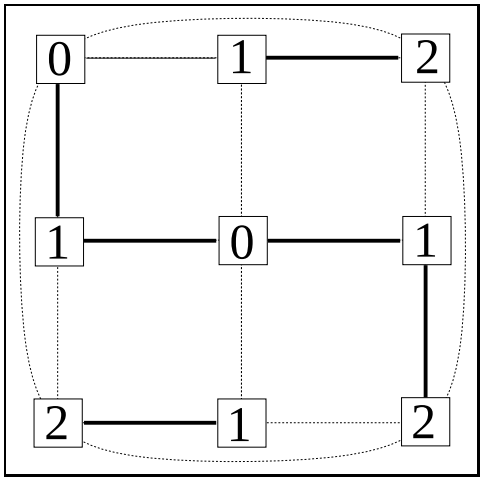

Figure 3.9: Etiquetage d'une grille 3-3 (2 étiquettes suffisent)

Le routage des paquets se fait de la façon suivante :

- A la source, un paquet est introduit dans un tampon de la classe 0
- A chaque fois qu'un paquet doit emprunter un arc en sens inverse de celui indiqué par le graphe associé à sa classe courante, au nœud suivant il passe dans la classe directement supérieure.

Le nombre K de classes de tampons nécessaires est borné par la plus grande distance que doit parcourir un paquet avant d'atteindre sa destination. Mais il devrait être inférieur puisque tant qu'un paquet reste dans la direction indiquée par le graphe associé à sa classe, il ne change pas de classe. Cependant, les auteurs ne donnent pas de méthode pour déterminer le nombre de classes ni la façon d'y associer les graphes orientés acycliques.

3.7.2.5 Méthodes des graphes virtuels et canaux virtuels

La méthode dite des graphes virtuels proposée par Jesshope et al. [Jesshope89a] consiste à diviser le réseau en un ensemble de graphes virtuels indépendants. Chaque graphe virtuel est orienté et acyclique et dispose de sa propre classe de tampons. Un paquet est introduit sur l'un des graphes virtuels et y reste jusqu'à destination. Il faut donc pouvoir trouver le bon graphe virtuel dès le départ. Le graphe des tampons dans cette technique est donc la réunion des graphes virtuels.

Pour ce qui est du nombre de classes de tampons par nœud, il n' y a pas de méthode générale pour déterminer un nombre minimun de graphes virtuels pour un réseau quelconque muni d'une fonction de routage. Les auteurs ont démontré la méthode sur des grilles en montrant comment elles peuvent être couvertes par quatre graphes virtuels, ainsi que des tores pour lesquels seize graphes virtuels sont nécessaires.

La technique des canaux virtuels [Dally86, Dally87] consiste à multiplexer sur un lien physique plusieurs canaux virtuels disposant chacun d'une classe de tampons propre. L'absence d'interblocage est assurée en construisant un graphe acyclique de canaux virtuels.

Dally et Seitz [Dally87] ont montré que des solutions utilisant un petit nombre de canaux virtuels par lien physique existent pour quelques réseaux réguliers. Ainsi 2 canaux suffisent pour les grilles ayant des connexions de rebouclage (grilles toriques), et 3 canaux sont nécessaire pour le CCC ("cube connected cycles"). Mais, comme pour la technique des graphes virtuels, dans le cas général il n'y a pas de de méthode pour déterminer le nombre minimum de canaux sur un lien physique.

3.8 Prévention de l'interblocage avec un nombre constant de classes de tampons

3.8.1 Les solutions dédiées

Parmi les solutions dédiées, on peut citer le "e-cube routing" [Dally86, Wittie81] utilsée pour les hypercubes binaires et les grilles non rebouclées. Elles sont en générale optimales mais ceci n'est pas une règle car pour certains réseaux, comme les grilles rebouclées (tores) et le CCC, les solutions optimales n'existent pas.

Si on considère un cas précis, en général on doit pouvoir trouver une solution avec un nombre constant de classes de tampons en appliquant certaines des méthodes ci-dessus [Alhafez91]. On profite alors des propriétés particulières du graphe régulier pour trouver une bonne solution. C'est le cas par exemple de la méthode des graphes virtuels appliquée aux grilles et aux tores [Jesshope89a].

3.8.2 Les solutions générales

Les solutions générales sont souvent basées sur l'existence d'un "réseau recouvrant" pour lequel on connait une solution. Nous entendons ici par réseau recouvrant, un réseau connexe ayant les mêmes nœuds que le réseau original, et dans lequel un lien n'existe que s'il existe dans le réseau original. Par exemple, une grille peut être utilisée comme réseau recouvrant d'un tore, mais dans le cas général il y'a qu'un réseau recouvrant qu'on peut trouver dans tout réseau connexe : l'arbre. Le routage sur un arbre recouvrant présente néanmoins les inconvénients suivants:

- la racine de l'arbre est un goulet d'étranglement,
- les chemins sont trop rallongés,

- les liens n'appartenant pas à l'arbre ne sont pas utilisés. Par exemple, dans le cas où tous les nœuds sont de degré quatre, la moitié des liens ne sont pas utilisés. Cette proportion augmente avec le degré des nœuds.

Pour un réseau quelconque, on ne peut pas trouver une fonction de routage possédant les trois caractéristiques absence d'interblocage, nombre de classes de tampons constant, et chemins les plus courts. L'absence d'interblocage étant nécessaire à la correction du routage, il y a un compromis à faire entre l'espace mémoire et la longueur des chemins.
Deux solutions originales ont été proposées pour ParX. La première, proposée par Gonzalez et décrite dans [Gonzalez91], est basée sur un arbre recouvrant et des règles d'utilisation des liens qui n'appartiennent pas à l'arbre (extra-liens). Nous avons proposé une deuxième solution que nous présentons au chapitre suivant.

Chapitre 4

Une nouvelle méthode de routage sans interblocage utilisant un nombre constant de tampons

4.1 Introduction

Dans le chapitre précédent nous avons dégagé les caractéristiques souhaitables pour une "bonne " fonction de routage. Ces caractéristiques sont idéales et ne peuvent pas être satisfaites toutes en même temps sauf pour quelques réseaux réguliers. Dans le cas général on doit donc chercher un compromis.

Le problème de la taille de l'information de routage sera abordé dans la dernière partie de ce rapport (chapitres 5, 6 et 7). Nous nous intéressons ici à l'abscence d'interblocage et aux tampons de stockage. Nous proposons une méthode pour le calcul de fonctions de routage sans interblocage et nécessitant un nombre constant de classes de tampons de stockage par nœud. Une première version de notre méthode, décrite dans [Mugwaneza90], a été utilisée dans MARC, un outil d'aide au placement et au routage pour réseaux de transputers dévéloppé à l'université de Berne [Boillat91]. La version implémentée dans PARX, décrite section 4.5.1, en est est une version optimisée. Nous considérons les réseaux de degré pair

ou ayant seulement deux nœuds de degré impair. La méthode ne nécessite alors qu'un seule classe de tampons par lien de sortie. Lorsque les hypothèses faites sur la parité du nombre de liens par nœud ne sont pas vérifiées, on peut s'y ramener en introduisant des canaux virtuels ; il suffit d'utiliser deux canaux virtuels sur chaque lien physique. On obtient un graphe virtuel dont les sommets sont de degré pair, et notre méthode revient à l'utilisation de deux classes de tampons par lien de sortie.

Les fonctions de routage issues de notre méthode assurent que les paquets sont délivrés à leurs destinataires. Peu de restrictions sont placées sur les chemins.

Dans les limites de ces restrictions, les paquets peuvent être acheminés sur les plus courts chemins. On peut aussi envisager un routage semi-adaptatif pour équilibrer le taux d'utilisation des liens.

L'idée est de choisir comme structure de routage un cycle ou un chemin[1] eulérien du graphe du réseau d'interconnexion. Supposant que les liens qui assurent le parcours du cycle dans les deux sens sont indépendants, on garantit par une partition appropriée de l'ensemble des communications susceptibles de s'établir, l'absence d'interblocage.

Le chapitre est organisé de la façon suivante : dans la section 4.2 nous rappelons quelques notions de la théorie des graphes et donnons quelques définitions et notations. La section 4.3 donne le principe de notre méthode sur un exemple très simple, celui de l'anneau bidirectionnel. La méthode est ensuite développée et évaluée dans la section 4.5.

4.2 Notations et définitions

Soit $G = (V, E)$ le graphe d'un réseau d'interconnexion ; V désigne l'ensemble des *nœuds* [2], et E l'ensemble des liens de communication. Les liens de communication sont supposés bidirectionnels. E désignera l'ensemble des arêtes (non orientées), et nous utiliserons A pour désigner l'ensemble des arcs de G. A chaque *arc* reliant un nœud v_1 à un nœud v_2 correspond un arc en sens inverse reliant v_2 à v_1. Les deux arcs correspondent à un lien bidirectionnel entre v_1 et v_2 et sont confondus en une seule *arête* du graphe non orienté.

Notation 4.1 *Dans ce chapitre,*

- *Nous noterons N le nombre de nœuds, et M le nombre de liens de communication. On a donc $N = |V|$ et $M = |E|$*
- *Un arc reliant deux sommets a et b est noté (a, b), a et b sont respectivement dits* extrémités initiale et terminale *de l'arc.*
- *Soit s un nœud de V, nous noterons $I(s)$ l'ensemble des arcs ayant s comme extrémité terminale (arcs entrants), et $O(s)$ l'ensemble des arcs ayant s comme extrémité initiale (arcs sortants).*

Le routage dans le réseau de graphe G consiste à définir au moins un chemin entre chaque paire de nœuds. Formellement nous considérerons les fonctions de routage qui sont sous la forme d'une application

$$\begin{array}{lll} R : & \mathrm{A} \times V \times V & \longrightarrow \mathcal{P}(A) \\ & (i, s, d) & \longmapsto S \end{array}$$

[1] Pour ne pas répéter à chaque fois cycle ou chemin, dans la suite nous parlerons de cycle

[2] Les termes *processeur* et *sommet* seront aussi utilisés indifféremment

où i est l'arc par lequel le paquet est arrivé à s, et S l'ensemble des arcs par lesquels le paquet peut sortir pour atteindre la destination d.

Définition 4.1 (table de routage) *: On appelle table de routage du nœud s par rapport à la fonction de routage R la restriction de R au nœud s, c'est à dire la matrice $(R(i,s,d)\ ;\ \ i \in I(s)$ et $d \in V)$.*

Nous avons vu au chapitre 3 que toute paire d'arcs dont l'extrémité terminale de l'un est extrémité initiale de l'autre correspond à une dépendance potentielle. La fonction de routage R ci-dessus choisit de matérialiser directement les dépendances autorisées. On définit la relation de dépendance associée à la fonction de routage de la façon suivante :

Définition 4.2 (relation de dépendance) *: Un arc i est dit dépendant d'un arc o par rapport à une fonction de routage R si et seulement si il existe un sommet d pour lequel $o \in R(i,s,d)$ où s est le sommet d'adjacence de i et o.*

Le graphe de dépendances est défini de la même façon qu'au chapitre 3.

Nous considérons maintenant le graphe non orienté, et définissons la notion de cycle ou chemin eulérien qui va nous servir comme support pour la définition de notre méthode de routage.

Définition 4.3 (cycle, chemin eulériens) *: Un cycle eulérien de G est un cycle passant une et une seule fois par chaque arête. Un chemin eulérien est un chemin passant une et une seule fois par chaque arête.*

Evidemment tout graphe G n'admet pas de cycle eulérien. Une condition nécessaire et suffisante pour ce faire est que tous les sommets de G soient de degré pair. Si dans G il y a exactement deux sommets de degré impair alors G admet un chemin eulérien.
A chaque graphe muni d'un cycle eulérien nous associons un anneau de la façon suivante :

Définition 4.4 (graphe associé à un cycle eulérien) *: Nous appelons graphe associé à un cycle eulérien du graphe $G = (V,E)$, le graphe orienté $G' = (V',E')$ construit de la façon suivante :*

- *on crée un nœud nommé 0 appelé "origine " du cycle, E' est initialisé à l'ensemble vide, et on initialise un compteur k à 1*
- *à chaque fois qu'on rencontre un nœud dans le parcours du cycle eulérien sur G et que l'on peut repartir du nœud ou que le nœud est différent de l'origine, on crée un nouveau sommet k dans G', on ajoute l'arc $(k-1,k)$ dans E' et on incrémente k de 1.*
- *si on ne peut pas repartir du nœud origine (cas d'un cycle eulérien), on ajoute l'arc $(k-1,0)$ dans E'.*

Sur le graphe G' d'un cycle eulérien de G, les nœuds de G peuvent donc apparaître plusieurs fois. G et G' sont reliés par les relations f et g suivantes :

$$\begin{array}{ll} f: & V' \longrightarrow V \\ & k \longmapsto f(k) \end{array}$$

$$\begin{array}{ll} g: & A \longrightarrow E' \\ & a \longmapsto g(a) \end{array}$$

où $f(k)$ est le sommet de V visité lors de la création de k, et $g(a)$ est l'arc de E' qui a été créé après le parcours de a. Nous dirons que k est une *occurrence* de $f(k)$ dans G' et que $g(a)$ est un *correspondant* sur G' de a.

Définition 4.5 (arcs directs, arcs indirects) *: Nous appelerons arcs directs de G, les arcs dont les correspondants sur G' sont dans le sens du cycle (c'est à dire relient un nœud k au nœud $(k+1) \bmod N$). Les arcs qui ne sont pas directs seront dits indirects.*

Les arcs de G' sont numérotés par leur extrémité initiale. Les arcs de G sont numérotés par le numéro de leur correspondant sur G'. Il y a donc une numérotation pour les arcs directs et une numérotation pour les arcs indirects. Deux arcs (direct et indirect) formant la même arête ont donc le même numéro.

Pour illustrer ces définitions et notations, considérons le réseau G de la figure 4.1. Les identifications des nœuds sont à l'intérieur des carrés qui les représentent, et sur chaque nœud figure les numéros des ports physiques qui le relient à ses voisins. G est un tore réalisé sur la machine supernode, où le contrôleur (nœud 17) a été inséré entre les nœuds 1 et 13. Le contrôleur est relié au nœud 0 qui permet la communication avec le monde extérieur. Sur la figure 4.1, nous avons matérialisé le parcours d'un chemin eulérien d'origine le nœud 17 en reliant à l'intérieur de chaque nœud traversé le port d'entrée au port de sortie. Ainsi, le chemin eulérien visite les sommets 17, 1, 2, 14, 10, etc.

Le graphe G' associé au chemin eulérien est donné par la figure 4.2. Les nœuds de G dont les sommets de G' sont des occurrences sont mis à côté. Les sommets de G' occurrence d'un même nœud sont reliés par une ligne pointillée. Ainsi, les sommets 0 et 33 de G' sont occurrences du nœud 17 ; de même, les sommets 15 et 11 sont occurrences du nœud 3. Nous avons omis l'orientation sur G', les arcs sont orientés dans le sens des aiguilles d'une montre et c'est ce sens qui définit les arcs directs sur G, le sens inverse définissant les arcs indirects.
La figure 4.3 donne la classification des arcs du nœud 6 ainsi que leur numérotation par le chemin eulérien.

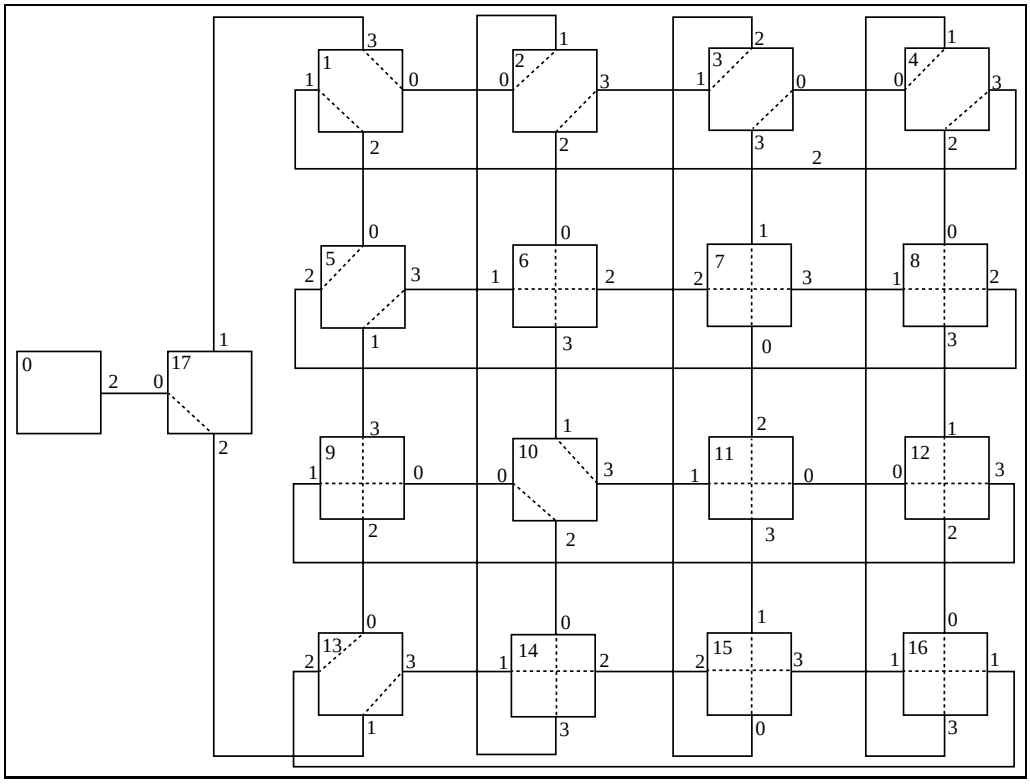

Figure 4.1: Un graphe d'interconnexion G et un chemin eulérien associé

4.3 Routage sans interblocage sur un anneau bidirectionnel

Considérons $G = (V, E)$ le graphe de l'anneau bidirectionnel de la figure 4.4. Nous avons vu au chapitre précédent que le routage sur les plus courts chemins dans G n'est pas sans interblocage. On peut obtenir une fonction de routage sans interblocage de la façon suivante :

- on choisit une arête $e \in E$ (par exemple celle reliant les nœuds 0 et 4) qu'on décide de ne pas utiliser ni dans un sens ni dans l'autre, et
- on route sur les plus courts chemins sur le nouveau réseau $G - \{e\}$

Il est immédiat que le routage sur $G - \{e\}$ est sans interblocage et ne nécessite qu'un tampon de stockage par lien de sortie. Par contre, ceci est au détriment de la longueur des chemins dont certains sont rallongés.

Toutefois, il n'est pas nécessaire d'interdire complètement l'utilisation de l'arête e. La fonction de routage sans interblocage sur un anneau peut subir une première amélioration en remarquant que les cycles dans le graphe de dépendances ne peuvent résulter que des chemins de longueur au moins égale à 2. L'arête e peut donc être utilisée mais seulement pour acheminer les paquets entre les deux sommets connectés. Les chemins à partir des nœuds 0 et 4 tiennent donc compte

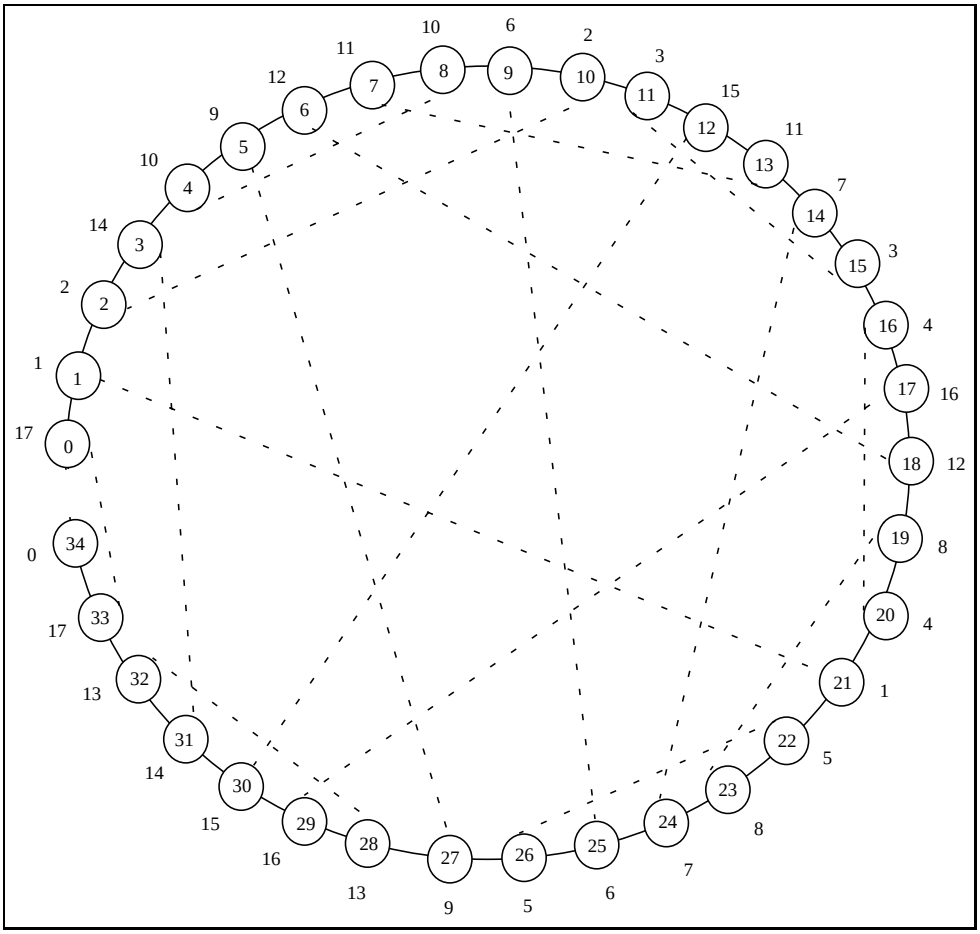

Figure 4.2: Graphe du chemin eulérien associé au réseau de la figure 4.1

Arcs entrants		Arcs sortants	
directs	indirects	directs	indirects
(10, 6) [8]	(2,6) [9]	(6,10) [8]	(6,2) [9]
(7,6) [24]	(5,6) [25]	(6,7) [24]	(6,5) [25]

Figure 4.3: Classification des arcs du nœud 6 (figure 4.1) en utilisant le chemin eulérien de la figure 4.2

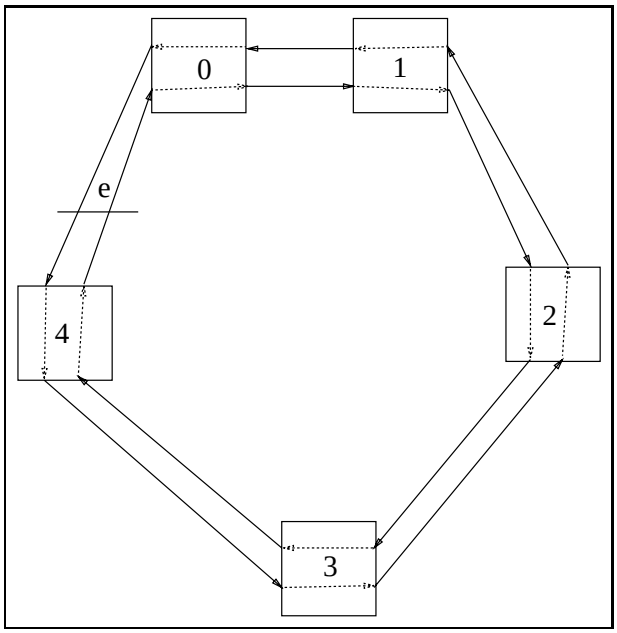

Figure 4.4: Routage sans interblocage sur un anneau bidirectionnel

non seulement de la destination, mais aussi de l'origine des paquets (locaux ou en transit).

Enfin, remarquons qu'une dépendance est une succession d'un arc entrant et d'un arc sortant, et que pour l'anneau il suffit d'interdire une dépendance dans chaque sens pour prévenir l'interblocage. On peut donc n'interdire que les dépendances $((4,0),(0,1))$ et $((1,0),(0,4))$[3]. La méthode de routage sans interblocage proposée dans la section suivante est basée sur cette notion de dépendances interdites.

4.4 Une nouvelle classe de méthodes de routage sans interblocage

Dans un réseau donné toute paire (*lien entrant*, *lien sortant*) est une dépendance potentielle notamment si la fonction de routage doit donner les chemins les plus courts. La méthode très simple que nous venons d'appliquer sur le cycle a consisté à éliminer des dépendances (2 dans le cas présent) de telle sorte que le graphe de dépendances résultant soit sans cycle. L'approche de construction de fonctions de routage sans interblocage que nous proposons part de cette idée. Le problème est alors de trouver un ensemble de dépendances à interdire pour avoir des fonctions de routage sans interblocage ayant de bonnes caractéristiques en ce qui concerne la longueur des chemins et/ou le taux d'utilisation des liens.

Nous commençons par formaliser le problème de routage correspondant à l'approche proposée. Notre solution est ensuite décrite dans la section 4.5. La méthode proposée est illustrée et évaluée à la section 4.5.4 pour la génération d'une fonction de routage déterministe.

4.4.1 Une formulation du problème de routage sans interblocage

Les notations et définitions utilisées ici sont celles de la section 4.2. Le problème du routage sans interblocage dans G peut être formulé de la façon suivante :

Problème 4.1 (P) *Trouver un ensemble F de dépendances potentielles à interdire tel que tout routage sur G qui n'induit pas de dépendance appartenant à F soit sans interblocage et ait de bonnes propriétés.*

Le problème P posé tel que nous venons de le faire n'est pas complètement défini. Pour affiner sa définition, un ensemble de paramètres doivent être fixés. Ils dépendent du type de fonction de routage désirée. En effet, suivant le mode de routage

[3]Evidement il n'est pas nécessaire que les deux dépendances interdites soient au même nœud.

utilisé (déterministe ou adaptatif) plusieurs instances du problème P peuvent être considérées.

Dans le cas où l'on s'intéresse à un routage adaptatif, il est important d'avoir le maximun de liberté dans le choix du lien de sortie. On peut alors s'intéresser à l'un des deux problèmes suivants, P2 charchant à exprimer une notion d'équilibre de charge.

Problème 4.2 (P1) *Trouver un ensemble minimun F de dépendances à interdire tel que toute fonction de routage qui n'utilise pas F soit sans interblocage.*

Problème 4.3 (P2) *Trouver un ensemble F composé d'à peu près le même nombre de dépendances potentielles sur tous les nœuds, et de cardinalité minimal tel que toute fonction de routage qui n'utilise pas F soit sans interblocage.*

Par contre, si on s'intéresse à un routage déterministe, le problème du routage sans interblocage peut être formulé de la façon suivante :

Problème 4.4 (P3) *Trouver un ensemble de dépendances potentielles F tel que le routage déterministe qui n'induit pas de dépendance appartenant à F et qui utilise les chemins les plus courts soit sans interblocage et fournisse des chemins de bonne qualité. La qualité des chemins étant mesurée par les critères tel que la distance entre les nœuds, et la charge des liens de communication* [4].

Ces problèmes sont des problèmes NP-complets. En effet on peut montrer qu'ils contiennent le problème de la recherche de chemins dans un graphe orienté muni de paires de sommets interdites, problème répertorié NP-Complet dans [Garey79].

Les solutions optimales ne sont donc pas envisageables, nous nous contenterons d'une méthode de détermination de l'ensemble F de dépendances à interdire qui assure que le graphe orienté résultant de la suppression des dépendances appartenant à F reste fortement connexe et que tout routage sur ce graphe est sans interblocage. Les autres critères sont évalués a posteriori par une mesure de la qualité d'un routage déterministe sur les chemins les plus courts.

Deux telles solutions ont été développées pour le noyau de communication de ParX. Gonzalez dans [Gonzalez91] propose une solution basée sur un arbre recouvrant pour déterminer l'ensemble F des dépendances interdites. Nous avons proposé une autre méthode basée sur un cycle eulérien qui interdit la même proportion de dépendances dans les nœuds.

Nous terminons cette section en présentant brièvement la méthode de Gonzalez. La section 4.5 est consacrée à la présentation et à l'évaluation de notre méthode de routage.

[4] Nous donnons une définition plus précise de la qualité des chemins à la section 4.5.4

4.4.2 Méthode basée sur un arbre recouvrant

A partir d'un nœud racine r, on construit un arbre recouvrant en suivant un parcours en largeur d'abord. A chaque nœud du réseau on associe son niveau qui correspond à la distance qui le sépare de la racine.

Définition 4.6 *Un arc est dit descendant s'il va d'un nœud de niveau i vers un nœud de niveau $i+1$, ascendant s'il va d'un nœud de niveau i vers un nœud de niveau $i-1$, horizontal s'il relie deux nœuds ayant le même niveau.*

La méthode de Gonzalez introduit les règles de routage suivantes :

Règle 4.1 *Un arc descendant ne peut dépendre d'un arc ascendant*

Règle 4.2 *Un arc horizontal ne peut dépendre d'un arc ascendant*

Règle 4.3 *Il ne peut y avoir de cycle d'arcs horizontaux dépendants.*

On montre aisement (cf. [Gonzalez91]) qu'en respectant les règles 4.1, 4.2 et 4.3 on peut toujours trouver un chemin entre chaque paire de nœuds, et que les fonctions de routage résultantes sont sans interblocage. Cette méthode présente néanmoins l'inconvénient de ne pas interdire uniformement les dépendances dans les nœuds.

4.5 Méthode du cycle eulérien

4.5.1 Principe de la solution

Soit G le graphe d'un réseau d'interconnexion. Nous supposons qu'au plus deux nœuds de G sont de degrés impairs, au besoin chaque arête du graphe est remplacée par deux arêtes virtuelles pour avoir un graphe dont tous les nœuds sont de degrés pairs. Au niveau du routage, cela revient à associer deux classes indépendantes de tampons à chaque lien de sortie.

Soit G' le graphe d'un cycle eulérien de G. Nous supposons que les arcs de G sont classés en arcs directs et arcs indirects et numérotés comme indiqué à la section 4.2. La méthode de routage que nous proposons consiste à observer les trois règles suivantes [5]:

Règle 4.4 *Aucun arc direct ne peut dépendre d'un arc indirect.*

Règle 4.5 *Aucun arc direct ne peut dépendre d'un arc direct de numéro inférieur.*

Règle 4.6 *Aucun arc indirect ne peut dépendre d'un arc indirect de numéro supérieur.*

Les règles 4.4, 4.5 et 4.6 définissent donc, à l'aide du cycle eulérien, un ensemble de dépendances interdites. Elles sont illustrées sur la figure 4.5 pour le nœud 6 de l'exemple de la section 4.2.

[5] Les règles présentées ici améliorent celles de [Mugwaneza90] qui n'autorisaient pas de dépendance (arc indirect, arc direct).

	(6,10)	(6,7)	(6,2)	(6,5)
(10,6)	-	interdite	autorisée	autorisée
(7,6)	interdite	-	interdite	autorisée
(2,6)	autorisée	interdite	-	autorisée
(5,6)	autorisée	autorisée	interdite	-

Figure 4.5: Dépendances interdites et dépendances autorisées au nœud 6 de figure 4.1 en utilisant le chemin eulérien de la figure 4.2

4.5.2 Correction de la méthode

Pour prouver la correction de notre méthode, nous montrons qu'en respectant les règles 4.4, 4.5, et 4.6 on peut toujours trouver un chemin entre n'importe quelle paire de processeurs et que les fonctions de routage produites sont sans interblocage.

Proposition 4.1 *L'application des règles 4.4, 4.5, et 4.6 laisse G fortement connexe.*

Demonstration : Soit s et d deux nœuds de G. Il nous faut montrer qu'on peut trouver un chemin reliant s à d qui ne contiene pas de dépendance interdite.
Soit k et l deux sommets de G' tels que $f(k) = s$ et $f(l) = d$ [6] . Deux cas sont possibles : $k < l$ ou $k > l$.

- 1) Supposons $k < l$. Considérons la suite $\mathcal{C}$ d'arcs directs de G correspondant aux arcs du chemin sur G' reliant k à l. Il est immédiat que $\mathcal{C}$ est un chemin reliant s à d et que $\mathcal{C}$ ne contient pas de dépendance interdite. Remarquons que $\mathcal{C}$ n'est pas forcément un chemin simple, c'est à dire qu'il peut passer plusieurs fois par un même nœud. Pour trouver un chemin simple il suffit d'éliminer les circuits les uns après les autres jusqu'à ce qu'il n'en reste plus.

- 2) Considérons maintenant le cas $k > l$. On procède de la même façon que pour le cas $k < l$ en prenant cette fois-ci un chemin $\mathcal{C}$ qui est une suite d'arcs indirects.

Donc dans tous les cas on trouve un chemin de s à d.
⊙

[6] Rappelons que $f(k)$ donne le sommet de G dont k est une occurrence

Proposition 4.2 *Toute fonction de routage qui n'induit pas de dépendance interdite par les règles 4.4, 4.5, et 4.6 est sans interblocage.*

Démonstration : Soit R une fonction de routage respectant les règles 4.4, 4.5, et 4.6. Nous devons montrer qu'il ne peut pas y avoir de cycle dans le graphe de dépendances associé à R.
D'après les règles de routage, R ne peut induire que des chemins de l'un des trois types suivants :

- une suite croissante d'arcs directs (règle 4.5),
- une suite décroissante d'arcs indirects (règle 4.6),
- une suite décroissante d'arcs indirects suivie d'une suite croissante d'arcs directs (règles 4.4, 4.5, et 4.6).

Supposons qu'il y ait un cycle dans le graphe de dépendances associé à R. Ce cycle ne peut pas être composé exclusivement d'arcs directs, car dans ce cas il y aurait un arc direct dépendant d'un autre arc direct de numéro inférieur ce qui violerait la règle 4.5. Ce cycle ne peut pas non plus être composé exclusivement d'arcs indirects car il y aurait violation de la règle 4.6. Le cycle comprend donc forcément des arcs directs et des arcs indirects, et donc forcément une dépendance composée d'un arc direct suivi d'un arc indirect. Or ce type de dépendance est interdite par la règle 4.4. Le graphe de dépendances associé à R ne peut donc pas contenir de cycle ; R est donc sans interblocage.
⊙

4.5.3 Evaluation de la méthode

Une façon d'apprécier la qualité de la méthode de routage proposée est d'évaluer le nombre de dépendances interdites. Le nombre de dépendances interdites dans chaque nœud est donnée par la proposition 4.3 suivante :

Proposition 4.3 *La proportion de dépendances autorisées par la méthode de routage définie par les règles de routage 4.4, 4.5 et 4.6 ci-dessus est de* $\frac{i-2}{2(i-1)}$ *pour le nœud origine du cycle eulérien et de* $\frac{i}{2(i-1)}$ *pour tout autre nœud, où* i *est le nombre de voisins du nœud.*

Demonstration : Le nombre total de dépendances possibles[7] dans un nœud de degré i est $i(i-1)$.

1) Cas d'un nœud non origine.
Soit s un nœud non origine du cycle eulérien et i son degré. Les dépendances autorisées se répartissent en trois groupes :

[7]Nous considérons le cas où l'on n'autorise pas qu'un paquet entré par un arc ressorte par l'arc en sens inverse

- Les dépendances composées de deux arcs directs : il y en a

$$\sum_{j=0}^{i/2} j = \frac{i}{4}(\frac{i}{2}+1)$$

- Les dépendances composées de deux arcs indirects : il y en a aussi $\frac{i}{4}(\frac{i}{2}+1)$
- Les dépendances composées d'un arc indirect suivi d'un arc direct. Il y en a $\frac{i}{2}(\frac{i}{2}-1)$

Le nombre total de dépendances autorisées dans un nœud de degré i qui n'est pas l'origine du cycle est donc

$$\frac{i}{4}(\frac{i}{2}+1) + \frac{i}{4}(\frac{i}{2}+1) + \frac{i}{2}(\frac{i}{2}-1) = \frac{i^2}{2}$$

et la proportion des dépendances autorisées par rapport au nombre total des dépendances possiles est de $\frac{i}{2(i-1)}$.

2) Cas du nœud origine du cycle.
Comme pour les autres nœuds les dépendances autorisées dans chaque nœud se répartissent en 3 groupes :

- Les dépendances composées de deux arcs directs : il y en a cette fois-ci

$$\sum_{j=0}^{i/2-1} j = \frac{i}{4}(\frac{i}{2}-1)$$

- Les dépendances composées de deux arcs indirects : il y en a aussi $\frac{i}{4}(\frac{i}{2}-1)$
- Les dépendances composées d'un arc indirect suivi d'un arc direct. Il y en a $\frac{i}{2}(\frac{i}{2}-1)$ comme pour tout autre nœud.

Le nombre total de dépendances autorisées au nœud origine du cycle est donc $i(\frac{i}{2}-1)$. Ce qui donne bien une proportion de $\frac{i-2}{2(i-1)}$.
$\odot$

Notre méthode interdit donc pratiquement la même proportion de dépendances dans tous les nœuds. L'impact de l'interdiction de ces dépendances sur la longueur des chemins est évalué ci-après sur une fonction de routage déterministe.

4.5.4 Application : une fonction de routage déterministe

Nous venons d'évaluer notre méthode de routage en calculant la proportion des dépendances autorisées par rapport au nombre total de dépendances. Nous allons maintenant évaluer l'évaluer par rapport à la longueur des chemins. Pour cela, nous considérons une fonction de routage déterministe qui route sur les chemins les plus courts respectant les règles de routage. Nous commençons par définir les paramètres que nous utilisons pour évaluer la longueur des chemins induits par la fonction de routage. Ces paramètres ont été calculés et comparés aux caractéristiques idéales du réseau.

4.5.4.1 Calcul d'une fonction de routage déterministe

Un fonction de routage sur les plus courts chemins issue de notre méthode de routage correspond tout simplement à la recherche des plus courts chemins dans un graphe orienté muni de paires interdites d'arcs (les dépendances interdites). Il suffit donc d'adapter les algorithmes classiques de calcul des plus courts chemins comme ceux décrites dans [Berge83].

4.5.4.2 Evaluation de la fonction de routage déterministe

En général les critères d'appréciation d'une fonction de routage sont en plus des critères de correction (validité, abscence d'interblocage), la longueur des chemins produits et la façon dont ces chemins sont sur l'ensemble des liens. Nous n'avons considéré ici que la longueur des chemins que nous évaluons par la notion d'élongation des chemins définie par Peleg et al. dans [Peleg89].
Nous commençons par donner quelques définitions.
Soit G le graphe d'un réseau d'interconnexion.

Notation 4.2 *Nous noterons $d_G(x, y)$ la distance entre les nœuds x et y, et D_G le diamètre de G.*

Définition 4.7 *(distance par rapport à une fonction de routage) : Soit R une fonction de routage, nous définissons la distance par rapport à R entre deux nœuds x et y et notons $d_{R,G}(x, y)$, la longueur d'un des plus courts chemins de x à y induits par R* [8].

Définition 4.8 *(élongation d'un chemin) : On définit l'élongation du chemin*[9] *entre deux nœud x et y par rapport à une fonction de routage R, et on note $\varrho_{R,G}(x, y)$, le rapport $d_{R,G}(x, y)/d_G(x, y)$ entre la distance par rapport à R et la distance sur le réseau original.*

L'élongation permet donc de comparer les chemins induits par une fonction de routage aux distances idéales sans contraintes.

Définition 4.9 *(diamètre d'une fonction de routage) : On appelle diamètre d'une fonction de routage R, et on note $D_{R,G}$, le maximum dans G des distances par rapport à R. $D_{R,G} = \max_{x,y}(d_{R,G}(x, y))$*

Définition 4.10 *(facteur d'élongation) : On appelle facteur d'élongation d'une fonction de routage R, et on note $\rho_{R,G}$ le maximun des élongations des chemins ($\rho_{R,G} = \max_{x,y}(\varrho_{R,G}(x, y))$).*

[8] $d_{R,G}$ n'est pas une distance au sens mathématique du terme. En effet, le graphe du réseau d'interconnexion étant considéré orienté et muni de paires interdites d'arcs, il n' y a aucune raison que $d_{R,G}(x, y) = d_{R,G}(y, x)$ pour tout couple (x, y)

[9] du plus court chemin

Définition 4.11 *(élongation moyenne d'une fonction de routage) : On appelle élongation moyenne d'une fonction de routage R, et on note $\overline{\rho_{R,G}}$, la moyenne arithmétique des élongations des chemins.*

Le diamètre d'une fonction de routage, ainsi que le facteur d'élongation sont des caractéristiques qui évaluent le pire des cas. Par contre, l'élongation moyenne reflète un comportement moyen de la fonction de routage.
Nous avons évalué les trois caractéristiques sur un jeu de tests préliminaire composé des grilles rebouclées à n dimensions et k nœuds par dimension (k-ary n-cubes). Les résultats sont données par les tables des figures 4.6, 4.7 et 4.8.
Dans les évaluations faites nous avons utilisé deux cycles eulériens différents. L'algorithme utilisé pour construire les deux cycles est celui décrit par Sakarovitch dans [Sakarovitch77]. Nous en donnons ci-dessous les grandes lignes.

- On part d'un sommet a quelconque, et l'on suit une chaîne C sans jamais utiliser deux fois la même arête jusqu'à ce que l'on ne puisse plus continuer.
- Itérer :
  - Si toutes les arêtes ont été visités on s'arrête.
  - Sinon on choisit un nœud sur C qui a encore aumoins une arête non encore visitée et l'on construit une nouvelle chaîne C'.
  - Insérer la chaîne C' dans la chaîne C.

Les deux cycles diffèrent par le choix de l'arête à suivre lors de la construction d'une chaîne. Le cycle R_1 choisit toujours l'arête ayant le plus petit numéro, tandis que le cycle R_2 choisit l'arête qui conduit au voisin ayant encore le plus grand nombre d'arêtes non encore visitées.

Notre méthode est comparée avec une fonction de routage sur un arbre recouvrant utilisant un parcours en largeur d'abord (colonne arbre), ainsi qu'à la méthode de Gonzalez basée sur le même arbre recouvrant (colonne arbre+).

La figure 4.6 montre le rapport $D_{R,G}/D_G$. La figures 4.7 et 4.8 donnent le facteur d'élongation et l'élongation moyenne.
Les évaluations faites sur le nombre de dépendances interdites montrent que le choix de l'origine du cycle n'est pas pertinent. Un problème qui cependant mérite d'être examiné est le choix du cycle eulérien car celui-ci semble avoir une influence sur les longueurs des chemins comme le montrent les évaluations. Intuitivement, le meilleur cycle serait celui qui répartirait uniformement les occurrences de chacun des sommets sur G', mais une tel cycle est difficile (sinon impossible) à construire car décrit par une propriété globale.

(n,k)	R_1	R_2	Arbre	Arbre+
(2,4)	1,2	1	1,75	1
(2,5)	1,50	1,50	2	1,50
(2,6)	1,33	1,33	1,83	1,33
(2,7)	1,50	1,60	2	1,67
(2,8)	1,37	1,50	1,87	1,50
(2,9)	1,50	1,62	2	1,75
(2,10)	1,40	1,50	1,90	1,60
(3,4)	1,17	1,33	1,83	1
(3,5)	1,33	1,67	2	1,50
(3,6)	1,22	1,33	1,89	1,33
(3,7)	1,33	1,44	2	1,67

Figure 4.6: Rapport diamètre du routage/ diamètre du réseau

(n,k)	R_1	R_2	Arbre	Arbre+
(2,4)	2	2	7	1
(2,5)	2,50	2	8	1,50
(2,6)	3	2	11	2
(2,7)	3,50	2,50	12	2,50
(2,8)	4	3	15	3
(2,9)	4,50	3,50	16	3,50
(2,10)	5	4	19	4
(3,4)	2	3	11	1
(3,5)	2,50	3,50	12	2
(3,6)	3	3	17	2
(3,7)	3,50	3,50	18	2,50

Figure 4.7: Facteur d'élongation du routage

(n,k)	R_1	R_2	Arbre	Arbre+
(2,4)	1,01	1,01	1,73	1
(2,5)	1,05	1,05	1,79	1,06
(2,6)	1,05	1,05	1,82	1,07
(2,7)	1,06	1,07	1,85	1,11
(2,8)	1,07	1,07	1,87	1,13
(2,9)	1,07	1,07	1,88	1,15
(2,10)	1,06	1,07	1,90	1,17
(3,4)	1,01	1,04	1,90	1
(3,5)	1,04	1,05	1,93	1,06
(3,6)	1,04	1,08	1,95	1,07
(3,7)	1,05	1,09	1,96	1,11

Figure 4.8: Elongation moyenne du routage

4.6 Conclusion

Dans ce chapitre nous avons proposé une nouvelle classe de méthode pour la génération de fonctions de routage sans interblocage. Notre méthode induit un nombre de tampons par nœud indépendant de la taille du réseau. Sauf pour des fonctions telles le "e-cube routing", adaptées à une connectique donnée, le routage sur les plus courts chemins et l'indépendance du nombre de tampons par processur de la taille du réseau sont en général incompatibles. Toutefois on aurait pu acheminer les messages sans restrictions sur les plus courts chemins. Mais alors la prévention de l'interblocage aurait coûté un tampon par nœud rencontré après chaque passage par l'arc interdit e. Cette pénalité ne pouvant être raisonnablement bornée, une alternative en train d'être évaluée est l'impact sur la longueur des chemins d'une introduction incrémentale de canaux virtuels.

Partie II

Compactage de l'information de routage

Présentation

Nous avons vu au chapitre 3 que le routage a deux principales composantes, l'acheminement et la fonction de routage. Chacune de ces composantes nécessite la ressource mémoire. Au chapitre 4, nous avons étudié les méthodes de prévention de l'interblocage dans le routage qui nécessitent un nombre de classes de tampons par processeur constant. Dans cette partie, nous considérons la deuxième composante et étudions le problème de la réduction de l'information nécessaire pour représenter la fonction de routage.

Chapitre 5

Introduction et état de l'art

5.1 Motivations

Pourquoi réduire l'information de routage?

La recherche de fonctions de routage pouvant être représentées par une information de routage compacte peut avoir plusieurs motivations. La plus significaticative à notre point de vue étant l'utilisation d'un processeur de communication qui réalise par le matériel toutes les fonctions du routeur. Il est alors nécessaire que l'espace mémoire nécessaire pour représenter la fonction de routage soit indépendante de la taille du réseau.
On pourrait imaginer un processeur de routage qui fait un accès mémoire pour consulter une fonction de routage, mais cette approche pour être utilisable nécessiterait une mémoire rapide. Bien que la technologie des mémoires ait fait des grands progrès on arrivera toujours à des situations où la taille des tables de routage dépassera la capacité d'une mémoire cache interne à un processeur.
L'utilisation de la mémoire externe augmente le nombre de broches du processeur de communication, celui-ci ayant une limite de l'ordre de la centaine, cette augmentation est au détriment du nombre de liens du processeur de communication. Nous nous intéresserons donc à des techniques de représentation des fonctions de routage qui nécessitent un espace mémoire indépendant du nombre de processeurs.

Quelle information de routage réduire?

Nous considérons les réseaux d'interconnexion dans lesquels chaque processeur de calcul est couplé avec un processeur de communication par le biais d'une voie de communication que nous appelerons *lien interne*, les processeurs de communication sont reliés entre eux suivant une topologie statique. Notre but étant l'intégration dans le matériel de toutes les fonctions du processeur de communication, seule l'information de routage nécessaire au processeur de communication est à compacter. Notamment, toute méthode de représentation de l'information de routage qui utilise des tables de routage générales au niveau du processeur de

calcul et une information synthétique de taille constante dans le processeur de communication répond à notre critère de compactage de l'information de routage. On pourrait dans un premier temps croire qu'on ne fait que transférer le problème du processeur de communication au processeur de calcul ; il n'en est rien car l'information non synthétique ne sera consultée qu'à l'initialisation de la communication et dans certains cas peut être incorporée dans les applications dans une phase de compilation et édition des liens.

Méthodes de réduction de l'information de routage

Une des premières méthodes de réduction de l'information de routage consiste à utiliser des fonctions de routage calculées. La fonction de routage est alors donnée par une expression mathématique qui est évaluée chaque fois qu'un paquet doit être retransmis. Dans certains cas, l'expression mathématique représentant l'information de routage est suffisamment simple pour être implémentée dans le matériel (ex : grilles multi-dimensionnelles et routage "e-cube " [Dally86]).

Les fonctions de routage calculées ne peuvent être trouvées que pour des réseaux réguliers. Dans le cas général, la fonction de routage est tabulée. L'information de routage non réduite consiste à maintenir dans chaque nœud une table de routage dans laquelle chaque nœud du réseau a une entrée qui donne, soit le premier lien sur le chemin entre les deux nœuds, soit tout autre type d'information sur ce chemin. Trois méthodes générales de réduction de l'information de routage ont été proposées dans la littérature : le routage hiérarchique, le routage par préfixes, et le routage par intervalles. Nous nous intéressons au routage par intervalles, la seule technique utilisant une information de routage dont la taille est indépendante du nombre de processeurs dans le réseau.

Organisation du chapitre

Dans ce chapitre nous présentons un état de l'art sur les techniques de génération de fonctions de routage avec une information de routage compacte. La section 2 présente trois techniques de compactage de l'information de routage : le routage hiérarchique, le routage par préfixes, et le routage par intervalles. Dans la section 3 nous nous intéressons au routage par intervalles et présentons les méthodes de génération de tables de routage par intervalles de la littérature.

5.2 Méthodes de compactage de l'information de routage

5.2.1 Routage hiérarchique

Principe

L'idée du routage hiérarchique proposé par Kleinrock et Kamoun dans [Kleinrock77] est de maintenir en chaque nœud une information de routage complète pour des nœuds qui lui sont "proches" (en termes de distance sur le réseau ou en utilisant tout autre critère), et une information synthétique pour les nœuds plus "éloignés". Ceci peut être réalisé en utilisant une table de routage ayant une entrée pour chaque nœud proche et une entrée par ensemble de nœuds éloignés. La taille des ensembles de nœuds éloignés pouvant croître au fur et à mesure que l'on s'éloigne du nœud.

La méthode proposée par Kleinrock et Kamoun consiste à organiser l'ensemble des nœuds du réseau sous forme hiérarchique. Chaque nœud est considéré comme étant un groupe de niveau d'hiérarchie 0 (0-cluster), les 0-clusters sont regroupés en 1-clusters, et plus généralement, les $(i-1)$-clusters sont regroupés en un certain nombre de i-clusters, ceci jusqu'à un niveau d'hiérarchie m pour lequel il n'y a qu'un seul m-cluster représentant tout le réseau.

La table de routage d'un nœud contient une entrée pour chaque nœud dans le même 1-cluster, une entrée pour chaque 1-cluster dans le même 2-cluster, et plus généralement, une entrée pour chaque $(k-1)$-cluster dans le même k-cluster $(k = 1, 2, \ldots, m)$.

La figure 5.1 montre un exemple de réseau hiérarchisé sur 3 niveaux ($m = 2$). La table de routage pour le nœud 2.2 de la figure 5.1 est donnée par la figure 5.2. On peut constater que 6 entrées sont nécessaires au lieu de 10 pour une table de routage classique.

Taille de l'information de routage

La taille des tables de routage dépend de la façon dont le réseau est organisé en clusters. Le problème étant de trouver une organisation du réseau qui entraîne une table de routage de taille minimum sur un nœud. Kleinrock et Kamoun [Kleinrock77] ont montré que pour un nombre m donné de niveaux d'hiérarchie, la table de routage a au moins $O(m \times N^{1/m})$ entrées, où N est le nombre de nœuds du réseau.
De plus, cette borne inférieure de [Kleinrock77] n'est atteinte que pour une certaine structure de l'hiérarchie de clusters dont les auteurs ne donnent aucune indication sur la façon de la construire.

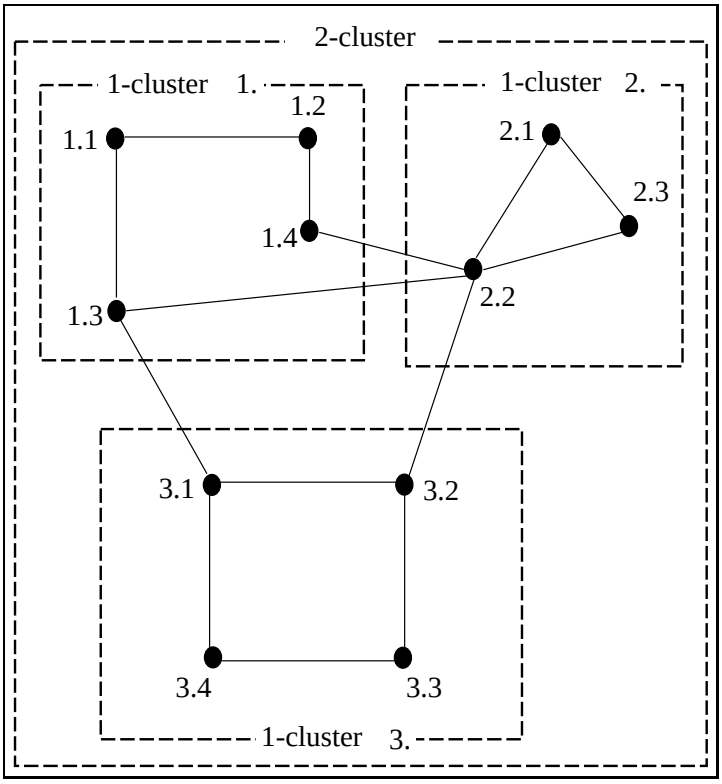

Figure 5.1: Un réseau hiérarchisé sur 3 niveaux

destination	nœud suivant
2.1	2.1
2.2	-
2.3	2.3
1.	1.3
2.	-
3.	3.2

Figure 5.2: Table de routage du nœud 2.2 pour le réseau de la figure 5.1.

Le problème de l'organisation du réseau en hiérarchie de clusters a été étudié par Bouloutas et Gopal dans [Bouloutas89]. Les critères de formation de clusters considérés sont la connectivité, le diamètre d'un cluster (distance maximale entre deux nœuds du cluster), le nombre de nœuds d'un cluster, et le nombre de connexions internes à un cluster. Bouloutas et Gopal montrent que pour des réseaux généraux, le problème de construction des clusters en respectant des contraintes utilisant un ou deux des ces critères est NP-complet.

La taille de la table de routage d'un nœud dépend donc du nombre de nœuds dans le réseau. La méthode de routage hiérarchique ne répond donc pas à notre critère de compactage de l'information de routage.

5.2.2 Routage par préfixes (Prefix-Routing)

Principe

Le routage par préfixes a été introduit par Bakker et al. [Bakker90] pour la minimisation de l'information de routage dans des réseaux pouvant évoluer dynamiquement par ajout ou suppression de nœuds ou de liens. L'idée du routage par préfixes est d'attribuer à chaque nœud x, une étiquette $\alpha(x)$ qui est un mot sur un alphabet Σ ($\alpha(x) \in \Sigma^*$, Σ contenant au moins deux symboles). Chaque lien est aussi étiqueté par un mot de Σ^* qui est un *préfixe* de quelques étiquettes de nœuds. Le routage se fait de la façon suivante : un paquet destiné au nœud x est transmis sur le lien dont l'étiquette est le plus long préfixe de $\alpha(x)$.

Bakker et al. [Bakker90] ont montré que tout réseau admet un routage par préfixes et proposent une méthode de construction de fonctions de routage par préfixes. Notons cependant que leur méthode ne considère pas le critère d'absence d'interblocage.

Taille de l'information de routage

D'un premier abord, puisqu'on associe une étiquette à chaque lien, la table de routage d'un nœud a une taille de l'ordre de son degré. Mais comme les étiquettes des nœuds peuvent devenir trop grandes (de l'ordre du diamètre du réseau), cette technique ne peut pas être considérée comme répondant à notre critère d'information de routage compacte.

5.2.3 Routage par intervalles

Principe

La table de routage classique consiste à représenter l'information de routage dans chaque nœud x par une relation $R_x \subset P \times L_x$ où P est l'ensemble des processeurs, et L_x l'ensemble des liens de communication du processeur x. Un couple $(d, l) \in$

R_x indique qu'au nœud x, le premier lien sur un chemin menant à d est l. Un lien utilisé pour plusieurs destinations va donc figurer plusieurs fois dans la table de routage représentant la relation R_x.

Une autre façon de voir l'information de routage est de considérer la relation ${R_x}^{-1}$, inverse de R_x. ${R_x}^{-1}$ associe à chaque lien $l \in L_x$ l'ensemble P_l des processeurs dont le chemin y menant a l comme premier lien. On peut donc représenter l'information de routage dans le nœud x par une table ayant autant d'entrées qu'il y a de liens dans L_x, l'entrée correspondant au lien l contenant l'ensemble P_l. Pour une destination d, la procédure de routage consiste alors à rechercher le lien i pour lequel $d \in P_i$ et à envoyer le paquet sur i.

Dans le cas général, cette deuxième représentation de la fonction de routage occupera sensiblement la même place que la représentation classique. Par contre il y a un surcoût pour la procédure de routage car cette fois, au lieu d'une simple consultation de table, il faut procéder à une recherche.
Un cas où l'on peut réduire la taille de l'information de routage et le temps de recherche pour la procédure de routage est celui où les processeurs peuvent être renommés en utilisant des noms appartenant à un ensemble totalement ordonné, et où chaque ensemble P_l est constitué d'éléments consécutifs. P_l est alors un intervalle et ses plus petit et plus grand éléments sont ses bornes. Dans ce cas pour représenter P_l il suffit de stocker ses bornes. On parle alors de *routage par intervalles.*

La recherche d'un élément dans l'intervalle se fait par simple comparaison avec les deux bornes. Comme le montre la figure 5.3 le calcul du lien de sortie est très rapide car les diverses comparaisons peuvent être effectuées en parallèle.

Taille de l'information de routage

La taille de l'information de routage nécessaire dans un nœud x est donc $O(\mid L_x \mid)$, c'est à dire de l'ordre du degré de x. La technique du routage par intervalles répond donc à notre critère de compactage de l'information de routage ; nous l'étudions dans la suite.

5.2.4 Problématique du routage par intervalles

Plusieurs façons de poser le problème du routage par intervalles peuvent être considérées :

Problème 5.1 (RI1) *: Etant donné une fonction de routage, trouver une méthode pour renommer les nœuds d'un réseau de telle sorte que pour chaque processeur x et chaque lien $l \in L_x$, l'ensemble P_l soit un intervalle.*

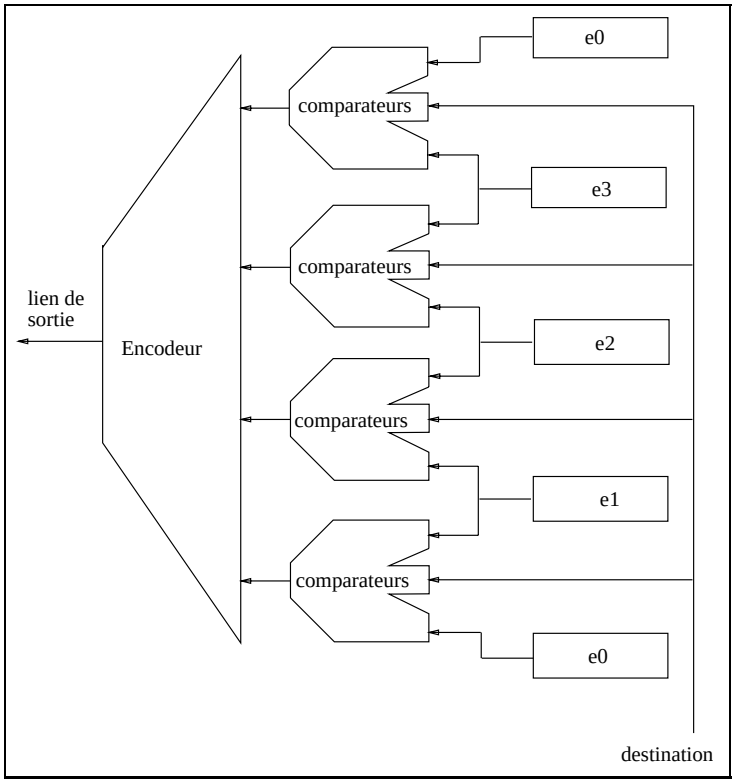

Figure 5.3: Calcul du lien de sortie pour une fonction de routage par intervalles

Si on sait résoudre le problème RI1, on peut utiliser tous les résultats sur les techniques de routage avec des tables de routage classiques. Malheureusement, le problème RI1 n'a pas toujours une solution. Par exemple, pour un tore de dimension supérieure à 4, si on considère une fonction de routage qui utilise les chemins les plus courts[1], on ne sait pas renommer les nœuds du tore de telle sorte que chaque ensemble P_l soit un intervalle [vLeeuwen87].
Dans le cas du tore, van Leeuwen et Tan [vLeeuwen87] ont montré que si on ne renomme pas les nœuds, deux intervalles sont nécessaires pour décrire le routage sur les plus courts chemins. Ceci conduit à considérer le problème RI2 ci-dessous.

Problème 5.2 (RI2) *: Etant donné une fonction de routage et un entier k, trouver une méthode pour renommer les nœuds du réseau de telle sorte que pour chaque processeur x et chaque lien $l \in L_x$, l'ensemble P_l soit une union d'au plus k intervalles.*

Le problème RI2 est un problème difficile car le nombre d'intervalles utilisé dépend à la fois de l'étiquetage des nœuds et de la fonction de routage considérée, et en général c'est la fonction de routage qui fixera la valeur de k.

L'approche du routage par intervalles que nous considérons consiste à partir d'un réseau et à rechercher des fonctions de routage ayant certaines propriétés

[1]Nous rappelons qu'une telle fonction de routage peut conduire à l'interblocage

de correction et pouvant être directement représentées par intervalles. Nous rechercherons essentiellement les propriétés d'absence d'interblocage et de validité. Nous nous intéressons donc au problème suivant :

Problème 5.3 (RI3) *: Etant donnés un réseau et un entier k, trouver une méthode pour renommer les nœuds du réseau et associer au plus k intervalles à chaque lien, de telle sorte que le routage induit par ces intervalles soit valide et sans interblocage.*

Le problème $RI3$ a des solutions, nous les présentons dans le reste de ce chapitre pour ce qui concerne l'état de l'art, et au chapire suivant pour notre contribution. Nous commençons par donner quelques définitions et notations.

5.2.5 Définitions et notations

La méthode de routage par intervalles a été introduite par Santoro et Khatib [Santoro82, Santoro85]. Les auteurs utilisaient de façon implicite la notion d'intervalle associé à un lien. van Leeuwen et Tan [vLeeuwen83, vLeeuwen86, vLeeuwen87] ont introduit le terme de routage par intervalles et clarifié les concepts. La plupart des définitions que nous donnons ci-dessous leur appartiennent.

Soit $G = (V, E)$ le graphe d'un réseau d'interconnexion. Notons $N = \mid V \mid$ le nombre de nœuds du réseau. Pour tout nœud $x \in V$, nous noterons L_x l'ensemble des liens bidirectionnels du processeur x, et d_x le nombre d'éléments de L_x. Soit $A = \{\alpha_0, \alpha_1, \ldots, \alpha_{N-1}\}$ un ensemble à N éléments muni d'une relation d'ordre $<$ telle que $\alpha_0 < \alpha_1 < \ldots < \alpha_{N-1}$.

Définition 5.1 (Intervalle) *: Soit a et b deux éléments de A. On appelle intervalle de borne inférieure a et de borne supérieure b et on note [a, b), l'ensemble défini comme suit :*

- *si* $a = b$ *:* $[a, b) = A$
- *si* $a < b$ *:* $[a, b) = \{\alpha_j / a \leq \alpha_j < b\}$
- *si* $b < a$ *:* $[a, b) = \{\alpha_j / a \leq \alpha_j\} \cup \{\alpha_j / \alpha_j < b\}$

Cette définition d'intervalle recouvre donc la notion classique d'intervalle mathématique sur un ensemble fini, à la quelle on rajoute une notion d'ordre cyclique[2].

Définition 5.2 (schéma d'étiquetage par intervalles) *: Un schéma d'étiquetage par intervalles*[3] $\mathcal{S}$ *d'un réseau* $G = (V, E)$ *est la donnée d'une numérotation des nœuds de G par les éléments de A et d'une numérotation des liens à chaque nœud telles que :*

[2]Le premier élément est successeur du dernier

[3]appelé aussi schéma d'étiquetage dans la suite

- *(a) chaque nœud x reçoit un numéro (étiquette) $\mathcal{S}_V(x) \in A$,*
- *(b) tous les nœuds reçoivent des étiquettes différentes* [4]*,*
- *(c) en chaque nœud x, chaque lien $l \in L_x$ reçoit un ensemble d'étiquettes $\mathcal{S}_E(l)$ qui est, soit vide, soit composé de une ou plusieurs étiquettes distinctes, et*
- *(d) en chaque nœud x, les étiquettes attribuées aux liens de L_x sont distinctes; c'est à dire, si $l_1 \in L_x$, $l_2 \in L_x$ et $l_1 \neq l_2$ alors $\mathcal{S}_E(l_1) \bigcap \mathcal{S}_E(l_2) = \emptyset$.*

Interprétation d'un schéma d'étiquetage en termes de routage par intervalles

Un schéma d'étiquetage par intervalles induit une fonction de routage par intervalles de la façon suivante. Soit $e_0, e_1, \ldots, e_{i-1}$ les étiquettes associées aux liens d'un nœud x. Sans perte de généralité on peut supposer $e_0 < e_1 < \ldots < e_{i-1}$. On associe les intervalles aux liens de la façon suivante :

- à chaque lien non étiqueté on associe l'ensemble vide.
- au lien ayant reçu les r étiquettes $e_{n_0}, \ldots, e_{n_j}, \ldots, e_{n_{r-1}}$, on associe les intervalles $([e_{n_j}, e_{(n_j+1) \bmod i}), 0 \leq j < r)$.

Les intervalles associés aux liens d'un nœud sont alors disjoints deux à deux et leur union couvre l'ensemble A des étiquettes.

Exemple

Considérons le réseau simple de la figure 5.4. L'ensemble A des étiquettes est $\{0, 1, 2, 3, 4\}$ La figure 5.4 donne deux exemples de schémas d'étiquetage, pour chaque lien les étiquettes et les intervalles correspondants sont indiqués.
Dans le cas du schéma d'étiquetage (a), un message émis par le nœud 2 à destination du nœud 1 passera par le nœud 4. Toutefois on peut remarquer que ce schéma d'étiquetage ne permet pas de déterminer un chemin entre toutes les paires de nœuds. Par exemple, tout paquet destiné au nœud 4 passe par le nœud 2 quelle que soit sa source et le nœud 2 envoie le paquet au nœud 3. Le paquet va donc décrire indéfiniment le cycle $0, 1, 2, 3$ sans jamais atteindre la destination. Le schéma d'étiquetage (a) ne décrit donc pas une fonction de routage correcte. Comme le montre la figure 5.4 (b), sur cet exemple simple on peut remédier facilement à cette situation, en donnant au lien de 2 vers 4 l'étiquette 4 au lieu de 1.

Tout étiquetage répondant à la définition 5.2 ne donne donc pas une fonction de routage valide. Dans la section suivante nous donnons un ensemble de définitions permettant de caractériser un schéma d'étiquetage.

[4] L'étiquette d'un nœud x étant unique, dans la suite nous la noterons x au lieu de $calS_V(x)$

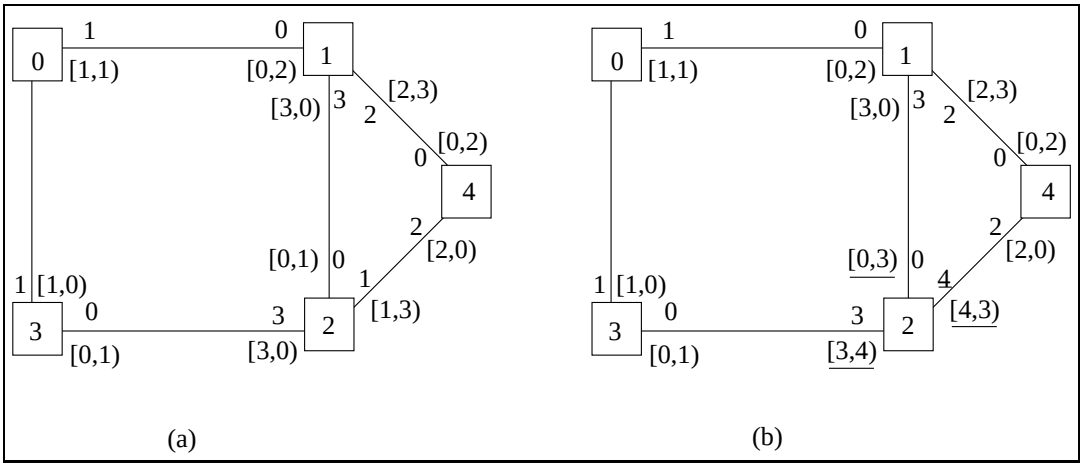

Figure 5.4: Exemples de schémas d'étiquetage par intervalles

5.2.6 Caractérisation des schémas d'étiquetage

Les définitions suivantes permettent de caractériser le routage induit par un schéma d'étiquetage.

Définition 5.3 (validité) *: Un schéma d'étiquetage est dit valide, si le routage induit est valide*

Définition 5.4 (absence d'interblocage) *: Un schéma d'étiquetage est dit sans interblocage, si le routage induit est sans interblocage.*

Définition 5.5 (optimalité) *: Un schéma d'étiquetage est dit optimal si le routage qu'il induit est optimal.*

Définition 5.6 (conservation du voisinage) *: On dit qu'un schéma d'étiquetage conserve le voisinage, si dans le routage qu'il induit tout nœud atteint tous ses voisins par des chemins de longueur* 1*.*

La propriété de conservation du voisinage est intéressante car elle assure que la fonction de routage garde une certaine localité du réseau. Ceci peut être utile pour d'autres types d'acheminement de messages que la communication point à point, la diffusion par exemple.

Les définitions qui suivent permettent de caractériser l'espace mémoire nécessaire pour représenter l'information de routage.

Définition 5.7 (étiquetage simple) *: Un schéma d'étiquetage est dit simple si chaque lien reçoit au plus une étiquette.*

Définition 5.8 (k-étiquetage) *: Un schéma d'étiquetage est dit k-étiquetage, si chaque lien recoit au plus k étiquettes.*

Définition 5.9 (étiquetage normal) *: Un schéma d'étiquetage pour lequel l'ensemble A des étiquettes est l'ensemble des N premiers entiers est dit normal.*

On peut utiliser des étiquettes autres que les N premiers entiers si, pour les besoins d'autres services système, ces étiquettes sont indispensables. En utilisant une table de correspondance maintenue en déhors du processeur de communication, on peut toujours se ramener à un étiquetage normal. Dans la suite nous ne considérons que les schémas d'étiquetage normaux.

Définition 5.10 (étiquetage séquentiel) *: Un schéma d'étiquetage est dit séquentiel si dans chaque nœud x étiqueté α_i, il y a un lien ayant l'étiquette $\alpha_{(i+1) \bmod N}$.*

Un étiquetage séquentiel permet de simplifier la structure du routeur en traitant comme les autres recherches le test pour savoir si le paquet est arrivé à destination. Ceci se fait simplement en considérant un lien interne auquel on associera toujours l'étiquette du nœud.

Les schémas d'étiquetage qui nous intéressent sont donc des k-étiquetages valides et sans interblocage avec k indépendant de la taille du réseau pour garder le critère de compactage de l'information de routage. Les autres critères ne seront pas néanmoins oubliés. Nous utiliserons le critère de conservation de voisinage et des mesures sur les longueurs des chemins pour comparer les schémas proposés.

5.3 Méthodes de routage par intervalles

5.3.1 Une famille de réseaux réguliers : grilles et hypercubes

La grille à n dimensions avec K_i éléments dans la dimension i est définie par la règle de connexion suivante : chaque nœud x est représenté par ses n coordonnées dans l'espace, $x = (x_0, x_1, \ldots, x_{n-1})$ avec $0 \leq x_i < K_i$. Dans la dimension i,

- si $0 < x_i < K_i$, x est connecté aux nœuds y et z définis par $y_j = z_j = x_j$ si $j \neq i$, $y_i = x_i - 1$, et $z_i = x_i + 1$
- si $x_i = 0$, x est connecté au nœud z tel que $z_j = x_j$ si $j \neq i$, et $z_i = x_i + 1 = 1$
- si $x_i = K_i - 1$, x est connecté au nœud y tel que $y_j = x_j$ si $j \neq i$, et $y_i = x_i - 1 = K_i - 2$

L'hypercube binaire est un cas particulier de la grille à n dimensions pour lequel $K_i = 2$ pour tout i.

La grille admet une fonction de routage optimale et sans interblocage, connue sous le nom du "routage e-cube " ou encore "routage en XY", qui consiste à

ordonner les dimensions, et à router tous les paquets dimension par dimension en suivant cet ordre.

Pour le routage "e-cube " on sait résoudre le problème RI2 ci-dessus. Le routage "e-cube " peut être répresenté par le schéma d'étiquetage $\mathcal{G}$ suivant proposé par van Leeuwen et Tan [vLeeuwen83] pour la grille à 2 dimensions et généralisé par May et Thompson dans [May90, May93].
L'ensemble A des étiquettes est donné par $A = \{0, \ldots, (\prod_{0 \leq i < n} K_i) - 1\}$. Un nœud x reçoit l'étiquette $\mathcal{G}_V(x)$ avec

$$\mathcal{G}_V(x) = \sum_{i=0}^{n-1} (x_i \prod_{j=0}^{i-1} K_j)$$

avec les conventions usuelles $\sum_{i=0}^{-1} x_i = 0$, et $\prod_{i=0}^{-1} x_i = 1$
S'ils existent,

- le lien reliant x à $(x_0, \ldots, x_{i-1}, x_i - 1, x_{i+1}, \ldots, x_{n-1})$ est étiqueté $\mathcal{G}_V(b)$ où $b_j = x_j$ si $i < j \leq n-1$, et $b_j = 0$ si $0 \leq j \leq i$.
- le lien reliant x à $(x_0, \ldots, x_{i-1}, x_i + 1, x_{i+1}, \ldots, x_{n-1})$ est étiqueté $\mathcal{G}_V(h)$ où $h_j = x_j$ si $i < j \leq n-1$, $h_i = (x_i + 1) \bmod K_i$, et $h_j = 0$ si $0 \leq j < i$.

La figure 5.5 illustre le schéma d'étiquetage $\mathcal{G}$ sur une grille 5×5. On peut facilement vérifier que cet étiquetage correspond à l'algorithme de routage "e-cube " par dimensions décroissantes [Sakho93].

5.3.2 Etiquetage pour réseaux généraux

5.3.2.1 Routage sur un arbre : méthode de Santoro et Khatib

Santoro et Khatib [Santoro82, Santoro85] sont les premiers à avoir introduit la notion de routage par intervalles bien que le terme n'apparait que dans les travaux de van Leuwen et Tan [vLeeuwen83, vLeeuwen86] qui ont suivi.

Santoro et Khatib utilisent le routage sur un arbre recouvrant des distances minimales (parcours en largeur d'abord) et donnent une méthode pour représenter ce routage en utilisant une information de routage compacte. Nous décrivons ici leur méthode en utilisant les concepts du routage par intervalles.

Etiquetage des nœuds

L'ensemble A des étiquettes est l'intervalle $[0, N)$ où N est le nombre de nœuds du réseau. L'étiquetage des nœuds est réalisé de la façon suivante :

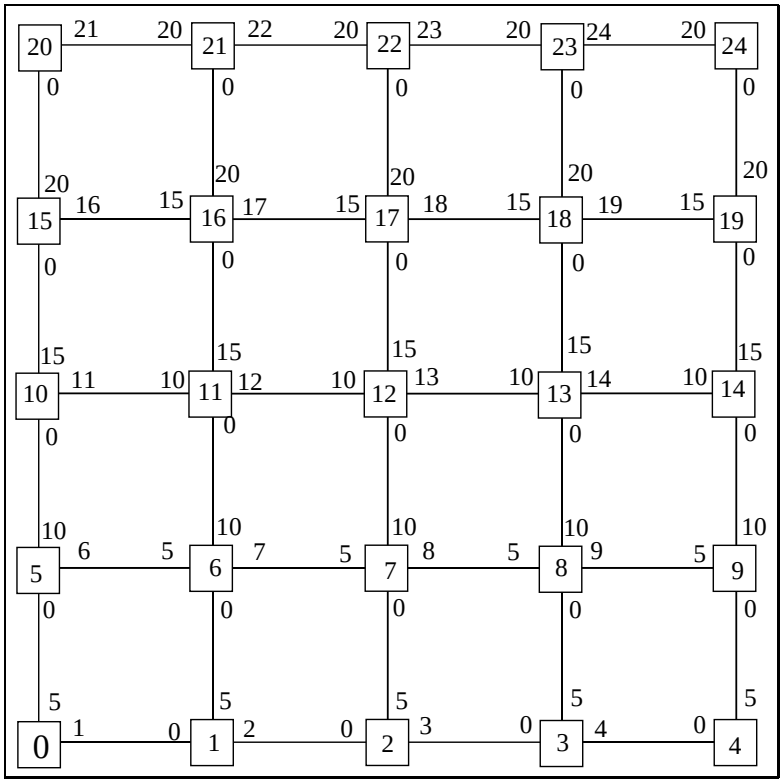

Figure 5.5: Schéma d'étiquetage $\mathcal{G}$ pour la grille 5×5

- à partir d'un nœud c centre[5] du réseau on construit un arbre recouvrant T en utilisant un parcours en largeur d'abord.
- on parcourt ensuite T en profondeur d'abord en attribuant une étiquette $\alpha(x)$ au nœud x la première fois que celui-ci est visité.
- les étiquettes sont attribuées dans l'ordre décroissant en commençant par $N-1$

Etiquetage des liens

A chaque nœud x, les liens sont étiquetés de la façon suivante :

- les liens qui n'appartiennent pas à l'arbre T ne sont pas étiquetés
- un lien reliant x à un de ses fils y, est étiqueté par $\min_{z \in T_y} \alpha(z)$ où T_y est le sous-arbre de T de racine y.
- le lien reliant x à son père, est étiqueté $\alpha(x)+1$

La figure 5.6 illustre la méthode de Santoro et Khatib, les liens de l'arbre T sont représentés en gras.

[5]Le centre d'un graphe est le nœud pour lequel le maximum de la distance par rapport à tout autre nœud du graphe est le plus petit (nœud d'écartement minimum).

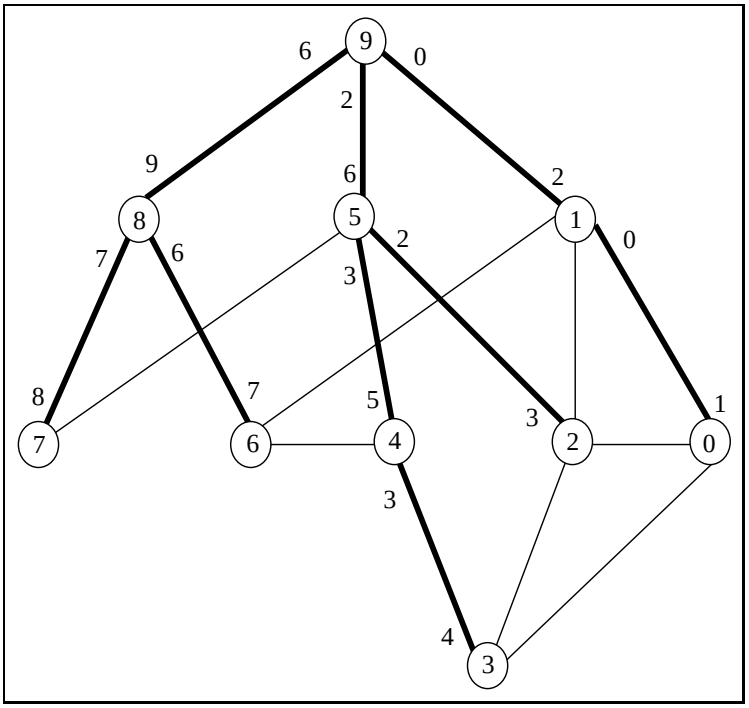

Figure 5.6: Un exemple d'étiquetage par la méthode de Santoro et Khatib

La méthode de Santoro et Khathib fournit des schémas d'étiquetage valides et sans interblocage mais elle présente les inconvénients du routage sur un arbre recouvrant : elle n'utilise que $N-1$ liens dans un réseau à N nœuds, et la racine ainsi que les nœuds qui lui sont proches sont des goulots d'étranglements. van Leeuwen et Tan essayent d'améliorer l'étiquetage de Santoro et Khatib en utilisant tous les liens du réseau[6] mais leur méthode peut conduire à des schémas d'étiquetage avec interblocage.

5.3.2.2 Méthode de van Leeuwen et Tan

On choisit un nœud racine c à partir duquel on construit un arbre recouvrant en suivant un parcours en profondeur d'abord. Les nœuds et les liens sont étiquétés pendant le parcours de la façon suivante :

- l'ensemble A des étiquettes est l'ensemble des N premiers entiers
- les étiquettes sont attribuées aux nœuds lors de la première visite et dans l'ordre croissant
- lors d'une visite du nœud x,
  - un lien vers un voisin y déjà étiqueté (y différent du père de x) reçoit l'étiquette $\alpha(y)$

[6]La méthode de van Leeuwen et Tan ne marche que quand il y a au plus un lien entre 2 nœuds. S'il y a plus d'un lien entre 2 nœuds, un seul d'entre eux est considéré.

  - un lien vers un voisin y non encore visité, est étiqueté par la valeur de la prochaine étiquette qui est attribuée à y immédiatement après.

- le lien d'un nœud x vers son père est étiqueté lors du retour en arrière en distinguant deux cas :
  - (a) si tous les nœuds n'ont pas encore été étiquetés, le lien de retour en arrière est étiqueté par la valeur de la prochaine étiquette à attribuer
  - (b) si tous les nœuds ont déjà été visités, le lien de retour en arrière est étiqueté 0, sauf si le nœud a déjà un lien étiqueté 0, auquel cas le lien de retour en arrière reçoit l'étiquette du père de x

La figure 5.7 illustre la méthode de Van Leeuwen et Tan, l'arbre recouvrant utilisé est représenté en gras.

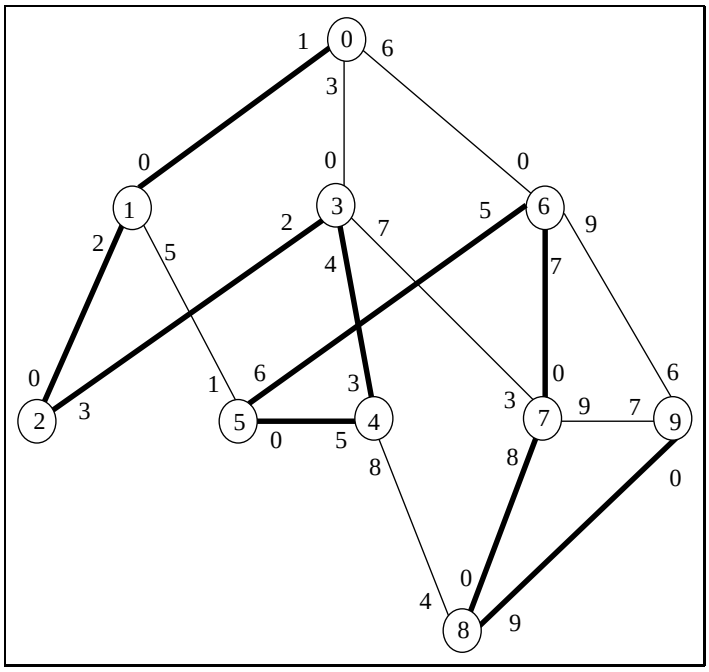

Figure 5.7: Un étiquetage par la méthode de Van Leeuwen et Tan

La méthode d'étiquetage de van Leeuwen et Tan induit un routage sur l'arbre en profondeur d'abord avec la possibilité de prendre certains raccourcis. Les auteurs ont montré qu'elle fournit un schéma d'étiquetage valide [vLeeuwen86]. Nous prouvons ci-dessous que la méthode ne fournit pas des schémas d'étiquetage sans interblocage.

Proposition 5.1 *La méthode de van Leeuwen et Tan n'est pas sans interblocage*

Démonstration :
Il suffit d'exhiber un exemple de réseau pour lequel le schéma d'étiquetage fournit par la méthode a un cycle dans le graphe des dépendances.

Pour cela considérons l'étiquetage de la figure 5.8. Cet étiquetage induit le cycle $((0,2),(2,1),(1,3),(3,5),(5,4),(4,0),(0,2))$.
Pour s'en convaincre il suffit de considérer les chemins de 0 vers 3, 1 vers 6, et 5 vers 2. Les dépendances induites par chacun des chemins sont indiqués par une flèche à coté du nœud.
⊙

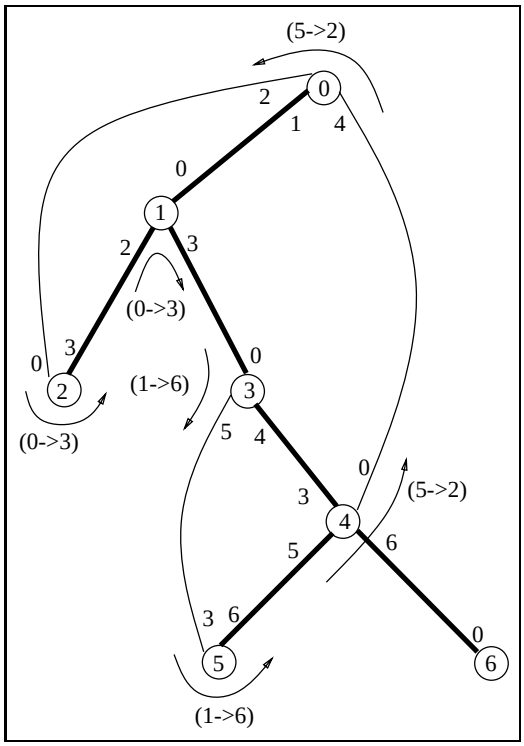

Figure 5.8: Exemple d'interblocage sur un réseau étiqueté par la méthode de van Leeuwen et Tan

5.4 Conclusion : bilan de l'état de l'art

Dans ce chapitre nous avons introduit le routage par intervalles, la seule technique générale de génération de tables de routage compactes. Malgré le fait que cette technique se prête bien à une intégration dans le matériel, peu de travaux ont été réalisés pour trouver des méthodes de génération de tables de routage par intervalles qui sont sans interblocage. La plupart des travaux n'ont considéré que le critère de validité.

Dans le chapitre suivant nous considérons le critère d'absence d'interblocage et proposons de nouvelles méthodes de génération de schémas d'étiquetage.

Chapitre 6

Nouveaux schémas d'étiquetage par intervalles

Dans ce chapitre nous présentons de nouvelles méthodes d'étiquetage par intervalles. Les schémas proposés sont valides et sans interblocage. La section 2 est consacrée au tore pour lequel nous proposons une méthode d'étiquetage qui utilise les liens de rebouclage. Dans la section 3 nous considérons les réseaux généraux et présentons une méthode d'étiquetage qui utilise chacun des liens au moins dans un sens. Les schémas générés par nos méthodes sont évalués et comparés avec les schémas présentés au chapitre précédent.

6.1 Un réseau régulier très utilisé : le tore

Le tore de dimension n est une grille de dimension n à laquelle on rajoute des liens de rebouclage dans chaque dimension.
Dans la dimension i, un nœud x est donc connecté à deux voisins y et z définis par $y_j = z_j = x_j$ si $j \neq i$, $z_i = (x_i + 1) \bmod K_i$, $z_i = (x_i + K_i - 1) \bmod K_i$ [1].

6.1.1 Schémas d'étiquetage

Si pour au moins une dimension i $K_i > 4$, van Leeuwen et Tan [vLeeuwen87] ont montré que même sans considérer le critère d'abscence d'interblocage, on ne peut pas trouver de schéma d'étiquetage simple, valide et optimal. Sakho et al. montrent dans [Sakho93] qu'on peut trouver un 2-étiquetage valide et optimal mais que ce schéma n'est pas sans interblocage.

Un schéma d'étiquetage simple et sans interblocage, mais non optimal existe pour le tore, il suffit d'utiliser le schéma d'étiquetage $\mathcal{G}$ de la grille présenté au chapitre précédent. Néammoins, cette solution a l'inconvénient de ne pas

[1] Pour simplifier, dans la suite nous noterons $x_i + 1$ et $x_i - 1$ à la place de $(x_i + 1) \bmod K_i$ et $(x_i + K_i - 1) \bmod K_i$.

utiliser les liens de rebouclage. Nous proposons ci-dessous un nouveau schéma d'étiquetage simple pour la grille qui utilise tous les liens. Les évaluations sur les longueurs des chemins montrent un gain notable apporté par l'utilisation des liens de rebouclage.

6.1.2 Un nouveau schéma d'étiquetage pour le tore

Soit le schéma d'étiquetage $\mathcal{T}$ défini de la façon suivante :
L'ensemble A des étiquettes est donné par $A = \{0, \ldots, (\prod_{i=0}^{n-1} K_i) - 1\}$.
Un nœud x reçoit l'étiquette

$$\mathcal{T}_V(x) = \sum_{i=0}^{n-1} (x_i \prod_{j=0}^{i-1} K_j)$$

Pour ce qui est de l'étiquetage des liens,

- le lien reliant x à $(x_0, \ldots, x_{i-1}, x_i + 1, x_{i+1}, \ldots, x_{n-1})$ est étiqueté $\mathcal{T}_V(h)$ où $h_j = x_j$ si $i < j \leq n-1$, $h_i = x_i + 1$, et $h_j = 0$ si $0 \leq j < i$.
- le lien reliant x à $(x_0, \ldots, x_{i-1}, x_i - 1, x_{i+1}, \ldots, x_{n-1})$ est étiqueté $\mathcal{T}_V(b)$ où $b_j = x_j$ si $i < j \leq n-1$, $b_j = 0$ si $0 \leq j < i$, $b_i = 0$ si $x_i \neq 0$ et $x_i \neq K_i - 1$, $b_i = \lceil K_i/2 \rceil$ si $x_i = 0$, et $b_i = \lceil K_i/2 \rceil - 1$ si $x_i = K_i - 1$.

La figure 6.1 illustre l'étiquetage $\mathcal{T}$ sur un tore 5×5. Les liens de rebouclage sur les lignes et les colonnes ne sont pas complètement tracés.

6.1.3 Preuve de correction du schéma $\mathcal{T}$

Nous allons montrer que le schéma d'étiquetage $\mathcal{T}$ est valide et sans interblocage. Pour cela nous commençons par énoncer et prouver les 3 résultats suivant.

Lemme 6.1 *Les intervalles induits par $\mathcal{T}$ sont des intervalles au sens mathématique du terme, c'est à dire que tout nœud $x \neq 0$ a un de ses liens étiqueté 0, et le nœud 0 a un lien étiqueté 1.*

Démonstration : Soit $x = (x_0, \ldots, x_{n-1})$. Si $x = 0$ le lien de 0 vers 1 a bien l'étiquette 1.
Si $x \neq 0$, soit d le plus grande dimension telle que $x_d \neq 0$.

- si $x_d \neq K_d - 1$ le lien de x vers y (avec $y_i = x_i$ si $i \neq d$ et $y_d = x_d - 1$) a l'étiquette 0.
- si $x_d = K_d - 1$ le lien de x vers z (avec $z_i = x_i$ si $i \neq d$ et $z_d = x_d + 1 = 0$) a l'étiquette 0.

$\odot$

Le lemme 6.2 assure qu'un paquet ne fera pas indéfiniment des allers et venues entre deux nœuds.

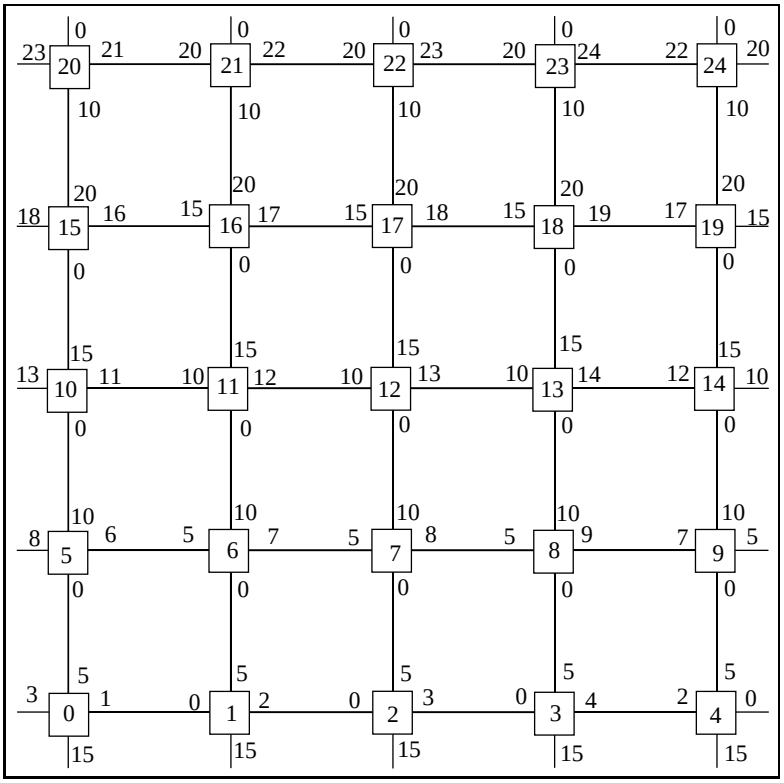

Figure 6.1: Etiquetage $\mathcal{T}$ d'un tore 5×5

Lemme 6.2 *Soit u et v deux nœuds voisins, l'intervalle associé au lien de u vers v et l'intervalle associé au lien de v vers u sont disjoints, ou leur intersection se réduit à l'un des deux nœuds u ou v.*

Démonstration :
Soit u et v deux nœuds voisins dans une certaine dimension d. Sans perte de généralité nous supposerons que $v_i = u_i$ si $i \neq d$ et $v_d = u_d + 1$.
Nous distinguons les trois cas : $u_d = K_d - 1$, $u_d = K_d - 2$, et $u_d < K_d - 2$.

Considérons d'abord le cas où $u_d = K_d - 1$.
On a $E_{uv} = (0, \ldots, 0, 0, u_{d+1}, \ldots, u_{n-1})$, et
$E_{vu} = (0, \ldots, 0, \lceil K_d/2 \rceil, u_{d+1}, \ldots, u_{n-1})$.
Nous montrons que les deux intervalles sont disjoints. Pour cela, comme $E_{uv} < E_{vu}$ et que d'après le lemme 6.1 tous les intervalles sont des intervalles mathématiques, il suffit d'exiber une étiquette E au nœud u telle que $E_{uv} <\mathrm{E} \leq E_{vu}$.
En considérant w le voisin de u tel que $w_d = u_d - 1$, on a
$E_{uw} = (0, \ldots, 0, \lceil K_d/2 \rceil - 1, u_{d+1}, \ldots, u_{n-1})$.
Et donc $E_{uv} <\mathrm{E}_{uw} < E_{vu}$.

Considérons maintenant les deux autres cas ($u_d \leq K_d - 2$).

- Si $u_d = K_d - 2$, on a $E_{uv} = (0, \ldots, 0, K_d - 1 = u_d + 1, u_{d+1}, \ldots, u_{n-1})$, et $E_{vu} = (0, \ldots, 0, \lceil K_d/2 \rceil - 1, u_{d+1}, \ldots, u_{n-1})$.

- Si $u_d < K_d - 2$, L'étiquette associé au lien de u vers v
  est $E_{uv} = (0, \ldots, 0, u_d + 1, u_{d+1}, \ldots, u_{n-1})$, celle associée au lien de v vers u est
  $E_{vu} = (0, \ldots, 0, 0, u_{d+1}, \ldots, u_{n-1})$.

Dans les deux cas on a donc $E_{uv} = (0, \ldots, 0, u_d + 1, u_{d+1}, \ldots, u_{n-1}) > E_{vu}$.
Nous distinguons deux sous cas suivant que $d = 0$ ou $d \neq 0$.

- (1) : $d = 0$
  On a $E_{uv} = v > E_{vu}$.
  En considérant w le voisin de v tel que $w_0 = v_0 + 1$ on a $E_{vw} = v + 1$. Donc l'intersection des deux intervalles de bornes inférieures E_{uv} et E_{vu} est, soit vide, soit réduite au seul nœud v.
- (2) : $d \neq 0$
  Considérons les voisins de v dans la direction $d - 1$. Nous distinguons trois sous-cas suivant la valeur de $v_{d-1} = u_{d-1}$.
  - (2.1) : $v_{d-1} \in]0, K_{d-1} - 1[$
    En considérant w voisin de v tel que $w_{d-1} = v_{d-1} - 1$ on a $E_{vw} = (0, \ldots, 0, u_d + 1, u_{d+1}, \ldots, u_{n-1})$, donc on a $E_{vu} < E_{vw} = E_{uv}$ et les deux intervalles sont disjoints.
  - (2.2) : $v_{d-1} = K_{d-1} - 1$
    On considère w le voisin de v tel que $w_{d-1} = v_{d-1} + 1$ et on a $E_{vw} = (0, \ldots, 0, u_d + 1, u_{d+1}, \ldots, u_{n-1}) = E_{uv}$. Donc $E_{vu} < E_{vw} = Euv$, et les deux intervalles sont disjoints.
  - (2.3) : $v_{d-1} = 0$
    Soit q la plus grande dimension inférieure à d telle que $v_q \neq 0$.
    * Si q n'existe pas, on considère w le voisin de v dans la direction 0 tel que $w_0 = v_0 + 1 = 1$ et on a $E_{vu} < E_{uv} = v < E_{vw} = v + 1$. Donc l'intersection entre les deux intervalles contient au plus le nœud v.
    * Si q existe nous distinguons les deux cas : $0 < v_q < K_q - 1$ et $v_q = K_q - 1$
      Si $0 < v_q < K_q - 1$ on considère w le voisin de v tel que $w_q = v_q - 1$,
      Si $v_q = K_q - 1$ on considère w tel que $w_q = v_q + 1 = 0$.
      Dans les deux cas on a $E_{vw} = (0, \ldots, 0, v_{q+1} = 0, \ldots, 0, u_d + 1, u_{d+1}, \ldots, u_{n-1}) = E_{uv}$. Donc $E_{vu} < E_{vw} = E_{uv}$ et les deux intervalles sont disjoints.

$\odot$

Lemme 6.3 *Soit x et y deux nœuds différents. Soit d la plus grande dimension pour laquelle $x_d \neq y_d$. Au nœud x les paquets destinés à y sont transmis sur l'un des deux voisins de x dans la dimension d.*

Démonstration : Il nous faut montrer que y se trouve dans l'un des deux intervalles associés aux liens de x vers ses voisins dans la dimension d.

On a $x = (x_0, \ldots, x_{d-1}, x_d, x_{d+1}, \ldots, x_{n-1})$ et $y = (y_0, \ldots, y_{d-1}, y_d, x_{d+1}, \ldots, x_{n-1})$ avec $y_d \neq x_d$.
Soit u et v les deux voisins de x dans la dimension d. Nous supposerons $u_i = v_i = x_i$ si $i \neq d$, $u_d = x_d + 1$ et $v_d = x_d - 1$. Suivant la valeur de x_d nous distinguerons les trois cas suivants : (1) $x_d = 0$, (2) $x_d = K_d - 1$, et (3) $x_d \neq 0$ et $x_d \neq K_d - 1$.

Cas (1) : $x_d = 0$

Les deux étiquettes associées aux liens de x dans la dimension d sont $E_{xu} = (0, \ldots, 0, 1, x_{d+1}, \ldots, x_{n-1})$ et $E_{xv} = (0, \ldots, 0, \lceil K_d/2 \rceil, x_{d+1}, \ldots, x_{n-1})$. Comme $y_d \neq 0$, y est supérieur ou égal à E_{xu} la plus petite des deux étiquettes. Pour montrer que y est dans l'un des deux intervalles associés à la dimension d, il suffit de montrer que tout autre étiquette est soit strictement supérieure à y, soit strictement inférieure à E_{xu} la plus petite des deux étiquettes dans la dimension d.

Soit donc i une dimension différente de d.

- Si $i < d$, une étiquette E d'un lien de x vers un de ses voisins dans la dimension i est de la forme $E = (e_0, \ldots, e_{d-1}, 0, x_{d+1}, \ldots, x_{n-1})$. Et donc $E < E_{xu} < E_{xv}$.
- Si $i > d$, suivant la valeur de x_i, nous distinguons 3 cas.
  - (1.1) $x_i = 0$: les deux étiquettes dans la dimension i sont alors $E_1 = (0, \ldots, 1, x_{i+1}, \ldots, x_{n-1})$ et $E_2 = (0, \ldots, \lceil K_i/2 \rceil, x_{i+1}, \ldots, x_{n-1})$. Comme $y_i = x_i = 0$, on a $E_2 > E_1 > y$.
  - (1.2) $x_i = K_i - 1$: les deux étiquettes dans la dimension i sont $E_1 = (0, \ldots, 0, x_{i+1}, \ldots, x_{n-1})$ et $E_2 = (0, \ldots, \lceil K_i/2 \rceil - 1, x_{i+1}, \ldots, x_{n-1})$. Comme $x_i = K_i - 1$, on a $E_1 < E_2 < E_{xu} < E_{xv}$.
  - (1.3) $x_i \neq 0$ et $x_i \neq K_i - 1$: les deux étiquettes dans la dimension i sont $E_1 = (0, \ldots, x_i + 1, x_{i+1}, \ldots, x_{n-1})$ et $E_2 = (0, \ldots, 0, x_{i+1}, \ldots, x_{n-1})$. Et donc $E_1 > y$ et $E_2 < E_{xu} < E_{xv}$.

Cas (2) : $x_d = K_d - 1$

Les deux étiquettes associées aux liens de x dans la dimension d sont $E_{xu} = (0, \ldots, 0, 0, x_{d+1}, \ldots, x_{n-1})$ et $E_{xv} = (0, \ldots, 0, \lceil K_d/2 \rceil - 1, x_{d+1}, \ldots, x_{n-1})$. Nous utilisons le même raisonnement que dans le cas (1).
y est supérieur ou égal à E_{xu} la plus petite des deux étiquettes. Pour montrer que y est dans l'un des deux intervalles associés à la dimension d, il suffit de montrer que tout autre étiquette est soit strictement supérieure à y, soit strictement inférieure à E_{xu} la plus petite des deux étiquettes dans la dimension d.

Soit i une dimension différente de d.

- Si $i < d$, une étiquette E de x vers un de ses voisins dans la dimension i est de la forme $(e_0, \ldots, e_{d-1}, K_d - 1, x_{d+1}, \ldots, x_{n-1})$. Comme $y_d < K_d - 1$, on a $E > y$. y ne peut donc pas appartenir à un intervalle de la dimension i.
- Si $i > d$ nous distinguons 3 cas suivant la valeur de x_i.
  - (2.1) $x_i = 0$: Les deux étiquettes dans la dimension i sont $E_1 = (0, \ldots, 1, x_{i+1}, \ldots, x_{n-1})$ et $E_2 = (0, \ldots, \lceil K_i/2 \rceil, x_{i+1}, \ldots, x_{n-1})$. Comme $y_i = x_i = 0$ on a $E_2 > E_1 > y$.
  - (2.2) $x_i = K_i - 1$: Les deux étiquettes dans la dimension i sont $E_1 = (0, \ldots, 0, x_{i+1}, \ldots, x_{n-1})$ et $E_2 = (0, \ldots, \lceil K_i/2 \rceil - 1, x_{i+1}, \ldots, x_{n-1})$. Comme $x_i = K_i - 1$ on a $E_1 < E_2 < E_{xu} < E_{xv}$.
  - (2.3) $x_i \neq 0$ et $x_i \neq K_i - 1$: Les deux étiquettes dans la dimension i sont $E_1 = (0, \ldots, x_i{+}1, x_{i+1}, \ldots, x_{n-1})$ et $E_2 = (0, \ldots, 0, x_{i+1}, \ldots, x_{n-1})$. Et donc $E_1 > y$ et $E_2 < E_{xu} < E_{xv}$.

Cas (3) : $x_d \neq 0$ et $x_d \neq K_d - 1$

Les deux étiquettes associées aux liens de x dans la dimension d sont $E_{xu} = (0, \ldots, 0, x_d + 1, x_{d+1}, \ldots, x_{n-1})$ et $E_{xv} = (0, \ldots, 0, 0, x_{d+1}, \ldots, x_{n-1})$. Soit i une

dimension différente de d.

- Si $i < d$, une étiquette E de x vers un de ses voisins dans la dimension i est de la forme $(e_0, \ldots, e_{d-1}, x_d, x_{d+1}, \ldots, x_{n-1})$. Suivant les valeurs de x_d et y_d on a :
  - $E > y$ si $x_d > y_d$, et
  - $E_{xv} < E < E_{xu} < y$ si $x_d < y_d$.

  Dans les deux cas y est dans l'un des deux intervalles de borne inférieure E_{xu} ou E_{xv}.
- Si $i > d$ nous distinguons 3 cas suivant la valeur de x_i.
  - (3.1) $x_i = 0$: Les deux étiquettes dans la dimension i sont $E_1 = (0, \ldots, 1, x_{i+1}, \ldots, x_{n-1})$ et $E_2 = (0, \ldots, \lceil K_i/2 \rceil, x_{i+1}, \ldots, x_{n-1})$. Comme $y_i = x_i = 0$, on a $E_2 > E_1 > y$.
  - (3.2) $x_i = K_i - 1$: Les deux étiquettes dans la dimension i sont $E_1 = (0, \ldots, 0, x_{i+1}, \ldots, x_{n-1})$ et $E_2 = (0, \ldots, \lceil K_i/2 \rceil - 1, x_{i+1}, \ldots, x_{n-1})$. Comme $x_i = K_i - 1$, on a $E_1 < E_2 < E_{xv} < E_{xu}$.
  - (3.3) $x_i \neq 0$ et $x_i \neq K_i - 1$: Les deux étiquettes dans la dimension i sont $E_1 = (0, \ldots, x_i{+}1, x_{i+1}, \ldots, x_{n-1})$ et $E_2 = (0, \ldots, 0, x_{i+1}, \ldots, x_{n-1})$. Et donc $E_1 > y$ et $E_2 < E_{xv} < E_{xu}$.

Donc dans tous les cas de figure on montre donc que y ne peut pas appartenir à un intervalle associé à un lien dans une dimension différente de d. Donc le lemme 6.2 est vrai.
⊙

Proposition 6.1 *Le schéma d'étiquetage $\mathcal{T}$ est valide.*

Démonstration : La validité de $\mathcal{T}$ découle directement des lemmes 6.2 et 6.3. En effet, d'après ces deux lemmes, $\mathcal{T}$ induit un routage par dimensions décroissantes (lemme 6.3) et sur une dimension un paquet ne peut pas retourner en arrière (lemme 6.2). Donc le paquet arrive à destination au bout d'un nombre fini de pas et $\mathcal{T}$ est valide.
$\odot$

Proposition 6.2 *Le schéma d'étiquetage $\mathcal{T}$ est sans interblocage.*

Démonstration : Il nous faut montrer que le graphe de dépendances associé à $\mathcal{T}$ n'admet pas de cycles.
Supposons que ce ne soit pas le cas.
Soit donc C un cycle dans le graphe de dépendances associé à $\mathcal{T}$. Comme le routage est fait par dimensions décroissantes, C ne peut contenir que des liens d'une et une seule dimension d. C est forcément de longueur K_d et implique tous les nœuds $(u^j, 0 \leq j \leq K_d - 1)$ avec ${u^j}_i = u_i$ si $i \neq d$ et ${u^j}_d = j$. Nous montrons que C ne peut exister car les dépendances $((u^1, u^0), (u^0, u^{K_d-1}))$ et $((u^{K_d-2}, u^{K_d-1}), (u^{K_d-1}, u^0))$ ne sont pas possibles.

Notons $E_{p,q}$ l'étiquette associée au lien de u^p vers u^q.
Considérons d'abord la dépendance $((u^1, u^0), (u^0, u^{K_d-1}))$.
Il nous faut montrer que l'intervalle associé au lien de u^1 vers u^0 et l'intervalle associé au lien de u^0 vers u^{K_d-1} sont disjoints. Nous avons
$E_{1,0} = (0, \ldots, 0, 0, u_{d+1}, \ldots, u_{n-1})$ et
$E_{0,K_d-1} = (0, \ldots, 0, \lceil K_d/2 \rceil, u_{d+1}, \ldots, u_{n-1})$
Considérons l'étiquette $E_{1,2}$, on a $E_{1,2} = (0, \ldots, 0, 2, u_{d+1}, \ldots, u_{n-1})$. Comme par hypothèse $K_d \geq 5$, on a $E_{1,0} <\mathrm{E}_{1,2} < E_{0,K_d-1}$. Et comme d'après le lemme 6.1 les intervalles induits par $\mathcal{T}$ sont des intervalles au sens mathématique du terme, l'intervalle associé au lien de u^1 vers u^0 et l'intervalle associé au lien de u^0 vers u^{K_d-1} sont disjoints, et la dépendance $((u^1, u^0), (u^0, u^{K_d-1}))$ ne peut pas exister.

Considérons maintenant la dépendance $((u^{K_d-2}, u^{K_d-1}), (u^{K_d-1}, u^0))$.
Nous avons :
$E_{K_d-2,K_d-1} = (0, \ldots, 0, K_d - 1, u_{d+1}, \ldots, u_{n-1})$ et
$E_{K_d-1,0} = (0, \ldots, 0, 0, u_{d+1}, \ldots, u_{n-1})$.
En considérant l'étiquette E_{K_d-1,K_d-2}, nous avons :
$E_{K_d-1,K_d-2} = (0, \ldots, 0, \lceil K_d/2 \rceil - 1, u_{d+1}, \ldots, u_{n-1})$.
Et donc $E_{K_d-1,0} < E_{K_d-1,K_d-2} < E_{K_d-2,K_d-1}$, et toujours d'après le lemme 6.1 l'intervalle associé au lien de u^{K_d-2} vers u^{K_d-1} et l'intervalle associé au lien de u^{K_d-1} vers u^0 sont disjoints.
Donc la dépendance $((u^{K_d-2}, u^{K_d-1}), (u^{K_d-1}, u^0))$ ne peut pas exister.
Donc le cycle C ne peut pas exister et le schéma d'étiquetage $\mathcal{T}$ est sans interblocage.
$\odot$

6.1.4 Qualité des chemins induits par $\mathcal{T}$

Pour apprécier l'apport de l'utilisation des liens de rebouclage nous avons comparé le schéma d'étiquetage $\mathcal{T}$, à la simple utilisation de l'étiquetage de la grille non rebouclée (étiquetage $\mathcal{G}$). Les paramètres mesurés sont le diamètre, le facteur d'élongation et l'élongation moyenne. La figure 6.2 montre une comparaison des deux étiquetages sur des tores comportant le même nombre k de nœuds dans chacune des n dimensions ($K_i = k$ pour tout i). Le tableau (a) donne le rapport entre le diamètre du routage et le diamètre du réseau, le tableau (b) le facteur d'élongation, et le tableau (c) l'élongation moyenne. On peut noter un gain appréciable apporté par l'utisation des liens de rebouclage. En effet, le rapport $D_{R,G}/DG$ est diminué de 40%, le facteur d'élongation de 60%, et l'élongation moyenne de 20%.

(n,k)	G	T
(2,5)	2	1,50
(2,6)	1,67	1,33
(2,7)	2	1,67
(2,8)	2	1,75
(2,9)	1,75	1,50
(2,10)	1,80	1,60
(3,5)	2	1,50
(3,6)	1,67	1,33
(3,7)	2	1,67

(a)

(n,k)	G	T
(2,5)	4	1,50
(2,6)	5	2
(2,7)	6	2,50
(2,8)	7	3
(2,9)	8	3,50
(2,10)	9	4
(3,5)	4	1,50
(3,6)	5	2
(3,7)	6	2,50

(b)

(n,k)	G	T
(2,5)	1,38	1,06
(2,6)	1,36	1,07
(2,7)	1,39	1,11
(2,8)	1,38	1,13
(2,9)	1,40	1,15
(2,10)	1,39	1,17
(3,5)	1,36	1,06
(3,6)	1,34	1,07
(3,7)	1,37	1,11

(c)

Figure 6.2: Performances de l'étiquetage $\mathcal{T}$ (colonne T) comparé à l'utilisation de l'étiquetage de la grille non rebouclée (colonne G). (a) : rapport $D_{R,G}/D_G$, (b) : $\rho_{R,G}$, (c) : $\overline{\rho_{R,G}}$

6.2 Une nouvelle méthode d'étiquetage pour réseaux généraux

La méthode de génération de tables de routage par intervalles proposée par Santoro et Khatib dans la section précédente est basée sur un arbre recouvrant. Elle a donc l'inconvénient de ne pas utiliser tous les liens du réseau. La méthode de van Leeuwen et Tan se propose de remédier à cet inconvénient en utilisant tous

les liens, mais nous avons montré que cette méthode peut conduire à des schémas avec interblocage.

Dans cette section nous proposons une méthode de génération de schémas d'étiquetage par intervalles qui combine les avantages d'un arbre recouvrant de parcours en largeur d'abord et l'utilisation de tous les liens. Le schéma $\mathcal{G}S$ proposé est simple et les liens n'appartenant pas à l'arbre sont utilisés dans un sens.
Nous commençons par présenter l'étiquetage des nœuds. La procédure d'étiquetage des liens est ensuite présentée et les schémas d'étiquetages générés par notre méthode sont prouvés valides et sans interblocage. La méthode est ensuite comparée à celles de Santoro et Khatib et van Leeuwen et Tan.

6.2.1 Etiquetage des nœuds

L'ensemble A des étiquettes est l'ensemble des N premiers entiers. L'étiquetage des nœuds est réalisé de la façon suivante :

- à partir d'un nœud c centre du réseau, construire un arbre recouvrant T en suivant un parcours en largeur d'abord,
- les étiquettes sont attribuées dans l'ordre croissant en commençant par 0,
- parcourir T en profondeur d'abord en attribuant une étiquette $\mathcal{G}S_V(x)$ au nœud x la première fois qu'il est accédé[2].

Notation 6.1 *Soit x un nœud de G, nous noterons T_x le sous-arbre de T de racine x, $max(x)$ la plus grande étiquette attribuée à un nœud dans T_x, et $P(x)$ le père de x dans l'arbre T.*

6.2.2 Etiquetage des liens

Soit $l = (u, v)$ un lien du réseau. Du côté du nœud u, l'étiquette associée à l est définie comme suit :

- si $u > v$ et l n'est pas une arête de T, $\mathcal{G}S_E(l) = v$
- si $u > v$ et l est une arête de T (on a alors $v = P(u)$), $\mathcal{G}S_E(l) = max(u) + 1$
- si $u < v$ et l est une arête de T (v est alors un fils de u), $\mathcal{G}S_E(l) = v$

La figure 6.3 donne l'étiquetage $\mathcal{G}S$ pour le réseau qui nous a servi d'exemple pour illustrer les méthodes de Santoro et Khatib, et van Leeuwen et Tan.
Les liens qui n'appartiennent pas à T ne sont donc utilisés que dans un sens, du nœud ayant l'étiquette la plus grande vers celui ayant l'étiquette la plus petite.

[2]Dans la suite l'étiquette d'un nœud étant unique, nous confondrons les étiquettes avec noms des nœuds et parlerons tout simplement de x au lieu de $\mathcal{G}S_V(x)$

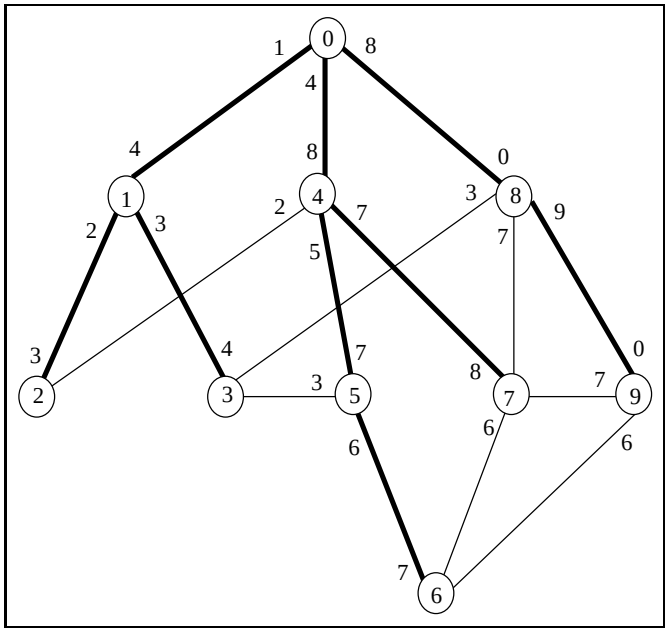

Figure 6.3: Un exemple d'étiquetage par la méthode $\mathcal{G}S$

6.2.3 Preuve de correction

Avant de montrer que GS_1 est valide et sans interblocage nous prouvons le lemme 6.4 ci-dessous.

Lemme 6.4 *Soit u un nœud de G et v un fils de u dans l'arbre T. L'intervalle associé au lien de u vers v est $[v, max(v)+1)$, et l'intervalle associé au lien de v vers u contient $[max(v)+1, 0)$ si $max(v)+1 \neq 0$.*

Démonstration :
Considérons d'abord l'intervalle associé au lien de u vers v.
Soit $f_1, \ldots, f_k$ les fils de u. Sans perte de généralité, on peut supposer $f_1 < f_2 < \ldots < f_k$. D'après la construction de l'arbre T et la règle d'étiquetage des nœuds on a :

- $u+1 = f_1 < max(f_1)+1 = f_2 < \ldots < max(f_{k-1})+1 = f_k$ et $max(f_k) = max(u)$.
- Les liens de u n'appartenant pas à l'arbre T peuvent être de deux types : (u, b) avec $b < u$ et (u, h) avec $h > max(u)$.

Un lien de type (u, b) est étiqueté b et un lien de type (u, h) n'est pas étiqueté. Il n'y a donc pas une étiquette de lien qui vient s'intercaler entre les éléments de la suite f_i. L'intervalle associé au lien de u vers f_i est donc $[f_i, max(f_i)]$.

Considérons maintenant l'intervalle associé au lien de v vers u.

On a $u = P(v)$. L'étiquette associé au lien de v vers $P(v)$ est $max(v) + 1$. Pour montrer que l'intervalle associé au lien de v vers $P(v)$ contient $[max(v) + 1, 0)$ il nous faut montrer que si $max(u) + 1 \neq 0$ tout autre étiquette attribuée à un lien de v vers un de ses voisins est inférieure à $max(u) + 1$.
Or ceci est garanti par la règle d'étiquetage. En effet, un lien de v vers $w \neq P(v)$

- est étiqueté w si $w < v$ ou $w \in T_u$,
- n'est pas étiqueté dans tous les autres cas.

Donc dans le cas où le lien a une étiquette celle-ci est bien inférieure à $max(u)+1$.
$\odot$

Proposition 6.3 *$\mathcal{G}S$ est valide*

Démonstration : Soit s et d deux nœuds différents. Il nous faut montrer que tout paquet envoyé de s à destination de d arrive après un nombre fini de pas. Suivant les valeurs relatives de s et d, on peut avoir l'un des trois cas suivants :

- (1) $s < d \leq max(s)$ (d est alors dans le sous-arbre T_s)
- (2) $max(s) < d < N - 1$
- (3) $0 \leq d < s$

Cas (1): $s < d \leq max(s)$

Soit v le fils de s tel que $v \leq d \leq max(v)$. D'après le lemme 6.4, s transmet le paquet vers v. Si $d = v$ le paquet est arrivé à destination et il est délivré, sinon on se retrouve dans le même cas avec cette fois-ci d dans le sous-arbre T_v. Comme T_v est strictement contenu dans T_s ce processus d'itération s'arrêtera après au plus $|T_v|$ pas.

Cas (2): $max(s) < d \leq N - 1$

D'après le lemme 6.4, s transmet le paquet vers $P(s)$ et on a $P(s) < s \leq max(s) < d \leq N - 1$. d est donc différent de $P(s)$ et on peut avoir deux cas : (2.1) $P(s) < d \leq max(P(s))$, et (2.2) $max(P(s)) < d < N - 1$.
Dans le cas (2.1), on retrouve le cas (1) et le paquet arrivera donc à destination. Dans le cas (2.2), le paquet est transmis sur le lien $(P(s), P(P(s)))$. Comme à chaque nœud intermédiaire t, $P(t) < t$, après un nombre fini de pas, on va se retrouver dans le cas (1).

Cas (3): $0 \leq d < s$

On distingue les deux sous-cas suivants :

- (3.1) : s a des voisins b_i ($b_1 < b_2 < \ldots < b_k < s$) et $b_1 < d < s$. Il existe alors une indice i tel que d appartient à l'intervalle associé au lien de s vers b_i, le paquet est retransmis sur ce lien. Dans b_i, soit $d = b_i$ et on a fini, soit $b_i < d < N - 1$ et on retrouve donc l'un des cas (1) ou (2) déjà traités.
- (3.2) : s n'a pas de voisins de type b_i ci-dessus ou $d < b_1$. Dans ce cas le paquet est retransmis vers $P(s)$. Dans $P(s)$, soit $d = P(s)$ et on a fini, soit $P(s) < d$ et on est donc dans l'un des cas (1) ou (2), soit $0 \leq d < P(s)$ et on est dans le cas (3). Dans ce dernier cas on atteindra la destination au bout d'un nombre fini de pas car à chaque fois l'étiquette du nœud intermédiaire décroit.

Dans tous les cas le paquet atteindra donc sa destination d.
⊙

Proposition 6.4 *$\mathcal{G}S$ est sans interblocage*

Démonstration : Il nous faut montrer que le graphe de dépendances de $\mathcal{G}S$ n'admet pas de cycles.

Supposons que ce ne soit pas le cas.
Soit donc $((v_1, v_2), (v_2, v_3), \ldots, (v_{q-1}, v_q), (v_q, v_1), (\mathrm{v}_1 v_2))$ un cycle dans le graphe de dépendances. On supposera $v_0 = v_q$ et $v_{q+1} = v_1$.
Comme le graphe G n'a pas de boucles, $v_j \neq v_{j+1}$ et il existe donc un indice i tel que $v_{i-1} < v_i$ et $v_i > v_{i+1}$. L'arête (v_{i-1}, v_i) est alors une arête de l'arbre T car tout autre arête (u, v) avec $u < v$ n'est pas étiquetée et ne peut pas être utilisée par le routage induit par $\mathcal{G}S$.
L'arête (v_i, v_{i+1}), elle, peut ne pas appartenir à l'arbre.

- Si (v_i, v_{i+1}) est une arête de l'arbre on a $v_{i+1} = v_{i-1} = P(v_i)$. Comme $\mathcal{G}S$ est valide, la dépendance $((P(v_i), v_i), (v_i, P(v_i)))$ ne peut pas exister.
- Si (v_i, v_{i+1}) n'est pas une arête de l'arbre, un nœud d atteignable en empruntant la dépendance $((v_{i-1}, v_i), (v_i, v_{i+1}))$ doit appartenir à l'intervalle $[v_i, max(v_i)]$ associé au lien de v_{i-1} vers v_i et à l'intervalle associé au lien de v_i vers v_{i+1}. Or ce dernier intervalle est de la forme $[v_{i+1}, \alpha)$ avec $\alpha \leq v_i$ si v_i a des fils, et $\alpha = max(v_i) + 1 = v_i + 1$ si v_i est une feuille de T. Comme $v_{i+1} < v_i$, les deux intervalles $[v_i, max(v_i)]$ et $[v_{i+1}, \alpha)$ sont disjoints[3].
  Donc la dépendance $((v_{i-1}, v_i), (v_i, v_{i+1}))$ ne peut pas exister.

Le graphe de dépendances ne peut donc pas admettre de cycle et $\mathcal{G}S$ est sans interblocage.
⊙

[3]En fait leur intersection peut aussi être réduit au seul nœud v_i

6.2.4 Qualité des schémas $\mathcal{G}S_1$

Comme pour le schéma d'étiquetage du tore, nous avons évalué le diamètre du routage, le facteur d'élongation et l'élongation moyenne. Les résultats sont présentés dans les tableaux de la figure 6.4. Les schémas obtenus en utilisant $\mathcal{G}S_1$ (colonne GS1) sont comparés avec ceux fournis par les méthodes de van Leeuwen et Tan (colonnes vLT), et Santoro et Khatib (colonnes SK). Les évaluations ont été faites sur les tores dont la structure parfaitement régulière fait que la qualité des schémas fournis par les méthodes utilisant un parcours en largeur d'abord (Santoro et Khatib, et $\mathcal{G}S_1$) ne dépend pas de l'ordre de parcours des listes d'adjacence. Par contre, pour la méthode de van Leeuwen et Tan qui utilise un parcours en profondeur d'abord, la qualité des schémas dépend de l'ordre de parcours. Les resultats présentés ici sont pour un parcours des voisins d'un nœud par ordre croissant des dimensions.

(n,k)	vLT	SK	GS1
(2,4)	2,50	1,75	1,75
(2,5)	4,50	2	2
(2,6)	4,67	1,83	1,83
(2,7)	6,67	2	2
(2,8)	6,75	1,87	1,87
(2,9)	8,75	2	2
(2,10)	8,80	1,90	1,90
(3,4)	7	1,83	1,83
(3,5)	15,50	2	2,17
(3,6)	19,11	1,89	1,89
(3,7)	32,78	2	2,11

(a)

(n,k)	vLT	SK	GS1
(2,4)	5	7	7
(2,5)	9	8	8
(2,6)	14	11	11
(2,7)	20	12	12
(2,8)	27	15	15
(2,9)	35	16	16
(2,10)	44	19	19
(3,4)	21	11	11
(3,5)	46.5	12	12
(3,6)	86	17	17
(3,7)	142,50	18	18

(b)

(n,k)	vLT	SK	GS1
(2,4)	1,29	1,73	1,55
(2,5)	1,55	1,79	1,71
(2,6)	1,75	1,82	1,75
(2,7)	2,02	1,85	1,79
(2,8)	2,25	1,87	1,81
(2,9)	2,55	1,88	1,83
(2,10)	2,81	1,90	1,84
(3,4)	1,64	1,90	1,78
(3,5)	2,42	1,93	1,87
(3,6)	3,41	1,95	1,89
(3,7)	4,97	1,96	1,91

(c)

Figure 6.4: Comparaison des schémas généraux d'étiquetage : longueur des chemins. (a) : rapport $D_{R,G}/D_G$, (b) : $\rho_{R,G}$, (c) : $\overline{\rho_{R,G}}$

Ces évaluations montrent que notre méthode et la méthode de Santoro et Khatib ont à peu près les mêmes caractéristiques en ce qui concerne les longueurs des chemins et, comme on pouvait s'y attendre, sont de loin meilleures que la méthode

de van Leeuwen et Tan. Notre méthode, à longueurs des chemins similaires, devrait mieux répartir les chemins sur les liens du réseau que la méthode de Santoro et Khatib

6.3 Conclusion

Dans ce chapitre nous nous sommes intéressés à de nouvelles méthodes de générations de schémas d'étiquetage par intervalles. La méthode pour le tore montre un gain notable apporté par l'utilisation des liens de rebouclage. La méthode générale, quoiqu'améliorant celle de Santoro et Khatib, manque les propriétes souhaitables suivantes :

- conservation du voisinage,
- utilisation de tous les liens dans les deux sens.

Le problème de savoir si pour un réseau quelconque, ces propriétés peuvent être satisfaites par un k-étiquetage (k indépendant de la taille du réseau) sans interblocage est un problème ouvert dans le cas général.

Dans le chapitre qui suit nous proposons une extension du routage par intervalles et une méthode de génération de tables de routage compactes qui sont sans interblocage et ont la propriété de conservation de voisinage.

Chapitre 7

Une nouvelle technique de représentation compacte de l'information de routage : le schéma d'étiquetage étendu (SEE)

7.1 Introduction

La notion de schéma d'étiquetage définie au chapitre 5 induit des fonctions de routage déterministes qui ne tiennent compte que de la destination pour choisir le lien de sortie. En outre, la question du nombre minimum d'étiquettes nécessaires pour représenter, par des intervalles, une fonction de routage donnée reste ouverte. En effet, nous avons vu au chapitre 6 que toutes les fonctions de routage, même si elles sont déterministes et n'utilisent que la destination, ne peuvent pas être représentées par des tables de routage compactes en utilsant des k-étiquetages avec k indépendant de la taille du réseau.

Nous proposons une extension de la notion de schéma d'étiquetage, le schéma d'étiquetage étendu (SEE), qui assure toujours une information de routage compacte pour le processeur de communication et qui couvre un spectre plus large de fonctions de routage que le schéma d'étiquetage par intervalles classique. Nous nous intéressons notamment à des fonctions de routage du même type que celle proposée au chapitre 4 qui tiennent compte du lien par lequel le paquet est entré pour déterminer le lien de sortie.

Le schéma d'étiquetage par intervalles classique n'utilise qu'une seule étiquette et un seul jeu d'intervalles par nœud. Le SEE étend la notion d'étiquetage en utilisant plusieurs étiquettes par nœud et en associant autant de jeu d'intervalles à un nœud qu'il a de connexions avec ses voisins. Chaque jeu d'intervalles est

alors utilisé pour le routage des paquets rentrés par le lien auquel il est associé. Le SEE permet donc de décrire des fonctions de routage qui utilisent le lien d'entrée et la destination d'un paquet pour déterminer le lien de sortie. En utilisant plusieurs étiquettes par nœud il permet aussi une certaine flexibilité en ce sens que les couches supérieures du système de communication peuvent influer sur le choix définitif d'un chemin entre deux nœuds (fonctions de routage semi-adaptatives).

Ce chapitre est organisé de la façon suivante :
Dans la section 7.2 nous introduisons la notion de SEE. La section 7.3 est consacrée aux méthodes de génération de SEE. Nous montrons notamment que la notion de SEE contient le routage par intervalles classique, et nous proposons une méthode générale de génération de SEE. Enfin dans la section 7.4 nous considérons le routage adaptatif. Nous y définissons la notion de SEE adaptatif et montrons comment générer un SEE adaptatif pour un réseau quelconque.

7.2 Schéma d'étiquetage étendu (SEE)

7.2.1 Définitions et notations

Nous reprenons la plupart des définitions et notations introduites aux chapitres 5 et 6 pour le réseau d'interconnexion et la notion d'intervalle. Rappelons que L_x désigne l'ensemble des liens de communication de x vers ses voisins, et ξ dénote le lien interne par lequel arrivent les paquets issus des processus locaux.

L'ensemble A des étiquettes est cette fois-ci un ensemble à Q éléments $A = \{\alpha_0, \alpha_1, \ldots, \alpha_{Q-1}\}$ avec $Q \geq N$. Comme au chapitre 5, A est muni d'une relation d'ordre total $<$ telle que $\alpha_0 < \alpha_1 < \ldots < \alpha_{Q-1}$.
Nous avons vu au chapitre 5 qu'on peut toujours se ramener à des schémas d'étiquetage normaux, dans ce chapitre nous prendrons donc $A = \{0, \ldots, Q-1\}$ l'ensemble des Q premiers entiers.

Notation 7.1 *: Nous noterons $\mathcal{P}(A)$ l'ensemble des parties de A, et $\mathcal{P}^+(A)$ l'ensemble des parties non vides de A.*

Définition 7.1 (numérotation multiple des nœuds d'un graphe) *: Soit $G = (V, E)$ un graphe, nous appelons numérotation multiple des nœuds de G, toute application f de V dans $\mathcal{P}^+(A)$ telle que si $x \neq y$ alors $f(x) \cap f(y) = \emptyset$.*
$f(x)$ est alors appelé l'ensemble des étiquettes du nœud x.

Une numérotation multiple associe donc une ou plusieurs étiquettes à chaque nœud.

Définition 7.2 (jeu d'étiquettes relatif à un nœud) *: Soit $G = (V, E)$ un graphe et x un nœud de G. Nous appelons jeu d'étiquettes relatif à x une ap-*

plication J de L_x dans $\mathcal{P}(A)$ qui à chaque lien $l \in L_x$ associe un ensemble $J(l)$ d'étiquettes tel que si $l_1 \neq l_2$ alors $J(l_1) \cap J(l_2) = \emptyset$

Un jeu d'étiquettes peut donc associer l'ensemble vide à un lien ; dans la pratique il y aura au moins un lien dont l'ensemble des étiquettes associé est non vide.

Définition 7.3 (schéma d'étiquetage étendu, SEE) *: Un schéma d'étiquetage étendu (SEE) du réseau G est la donnée de*

- *une numérotation multiple des nœuds de G, et*
- *pour chaque nœud x, $\mid L_x \cup \{\xi\} \mid$ jeux d'étiquettes relatifs à x associés chacun à un lien.*

Notation 7.2 *Soit S un SEE du graphe $G = (V, E)$ et x un nœud de G. Nous noterons $S_V(x)$ l'ensemble des étiquettes de x dans la numérotation multiple de S, et $S_E(x, l_1, l_2)$ l'ensemble des étiquettes associées au lien $l_2 \in L_x$ par le jeu d'étiquettes du lien $l_1 \in L_x \cup \{\xi\}$.*

7.2.2 exemple

Considérons le SEE S représenté sur la figure 7.1. Sur cette figure nous avons représenté un graphe G à 5 sommets a, b, c, d et e. Les étiquettes des nœuds sont notées à l'intérieur des carrés représentant les nœuds, et les jeux d'étiquettes associés aux liens sont donnés par les tables à coté. Dans la table d'un nœud, chaque colonne représente le jeu d'étiquettes associé au lien indiqué dans l'entête de la colonne par le nom du nœud voisin.
L'ensemble des étiquettes est $A = \{0, \ldots, 6\}$.
La numérotation multiple des sommets est donnée par :

$$S_V(a) = \{4\}$$
$$S_V(b) = \{0, 3\}$$
$$S_V(c) = \{1\}$$
$$S_V(d) = \{2, 6\}$$
$$S_V(e) = \{5\}$$

Les jeux d'étiquettes associés au nœud e sont :
- lien (e, a) :

$$S_E(e, (e, a), (e, a)) = \{0, 2\}$$
$$S_E(e, (e, a), (e, d)) = \{1, 6\}$$

- lien (e, d) :

$$S_E(e, (e, d), (e, a)) = \{0\}$$
$$S_E(e, (e, d), (e, d)) = \{3\}$$

- lien interne ξ :

$$S_E(e, \xi, (e, a)) = \{0, 2\}$$
$$S_E(e, \xi, (e, d)) = \{1, 6\}$$

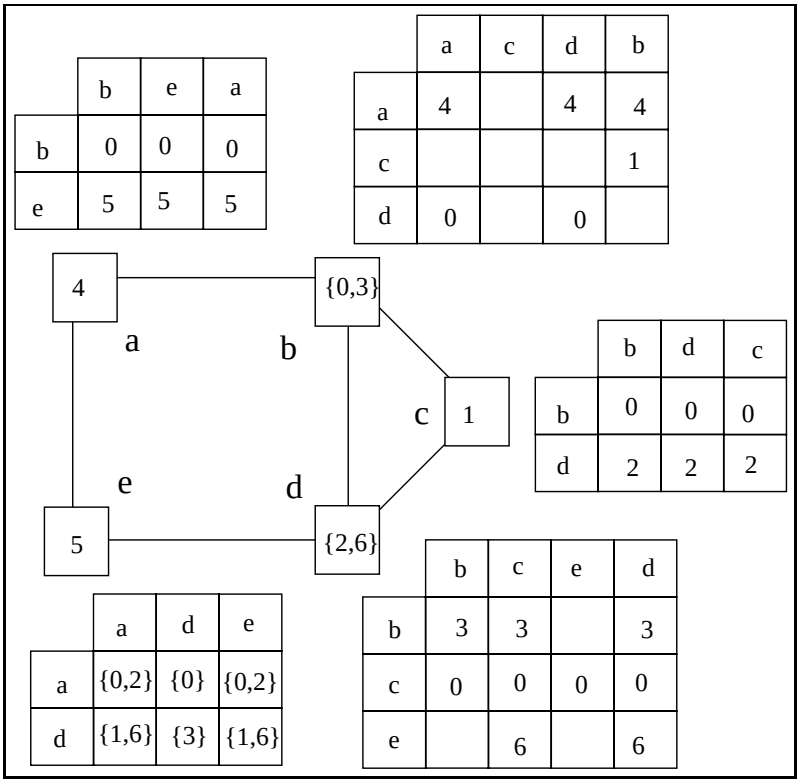

Figure 7.1: Un exemple de SEE (S)

7.2.3 Interprétation d'un SEE en termes de routage

La notion de SEE permet de définir de façon naturelle une fonction de routage par intervalles de la façon suivante :

- Chaque entête de paquet contient une étiquette de la destination,
- On associe les intervalles à un jeu d'étiquettes de la même manière qu'au chapitre 5. La figure 7.2 donne les intervalles associés aux liens des nœuds a et e par le SEE S de la figure 7.1.
- Pour le routage d'un paquet on procède de la même façon que pour le schéma d'étiquetage classique, mais cette fois-ci on utilise les intervalles correspondant au lien interne si la source du paquet est le nœud courant, et les intervalles associés au lien par lequel le paquet est entré si le paquet est en transit.
- Chaque nœud reconnaît les étiquettes qui lui ont été attribuées et donc les paquets qui lui sont destinés.

Pour illustrer la procédure de routage, considérons sur la figure 7.1 le cas où le nœud a veut envoyer un paquet vers d. a chosit une étiquette de d, 6 par exemple, et consulte le jeu d'étiquettes associé au lien interne ξ (colonne a). Il trouve que l'étiquette 6 est dans l'intervalle $[5, 0)$ associé au lien vers e (voir figure 7.2), et le paquet est envoyé vers e. Pour retransmettre le paquet, e utilise le jeu

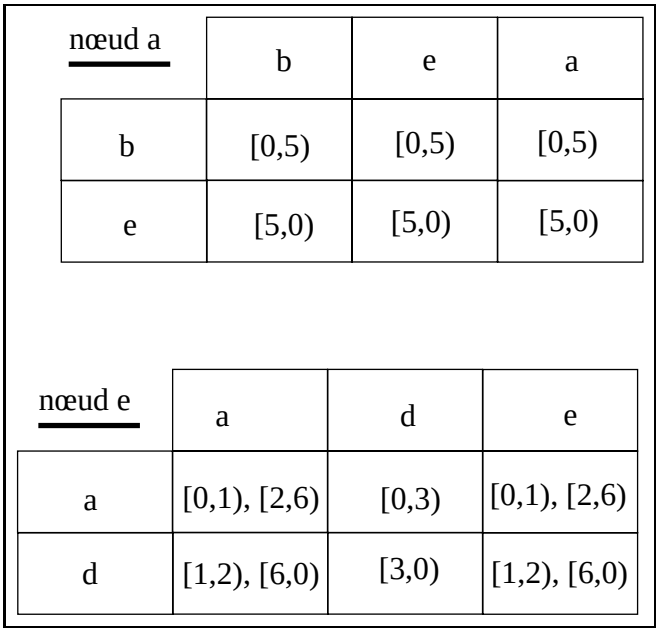

Figure 7.2: Intervalles associés aux liens des nœuds a et e dans le SEE S de la figure 7.1

d'étiquettes du lien par lequel le paquet est entré (colonne a). L'étiquette 6 est contenue dans l'un des deux intervalles [6,0) et [1,2) associés au lien vers d. Le paquet est donc retransmis vers d qui reconnaîtra 6 comme étant l'une de ses étiquettes et délivrera le paquet.

7.2.4 Caractérisation d'un SEE

Un SEE induit une fonction de routage, nous reprendrons donc les différentes propriétés pour caractériser une fonction de routage. Comme pour un schéma d'étiquetage classique on distinguera les propriétés relatives à la qualité des chemins(validité, optimalité, absence d'interblocage et conservation de voisinage) de celles qui caractérisent la représentation de l'information de routage.
Nous rappelons d'abord les 4 propriétés relatives à la qualité des chemins.

Définition 7.4 (validité) *: Un SEE est dit valide s'il induit une fonction de routage valide.*

Définition 7.5 (optimalité) *: Un SEE est dit optimal s'il induit une fonction de routage optimale.*

Définition 7.6 (absence d'interblocage) *: Un SEE est dit sans interblocage s'il induit une fonction de routage sans interblocage*

Définition 7.7 (conservation de voisinage) *: Un SEE conserve le voisinage s'il induit une fonction de routage qui conserve le voisinage.*

Pour caractériser la taille de l'information de routage nécessaire à la représentation de la fonction de routage, les notions de SEE simple et de k-SEE sont définies de la même manière qu'au chapitre 5. Nous rajoutons une propriété spécifique aux SEE, la valence.

Définition 7.8 (valence d'un SEE) *: La valence d'un SEE est le nombre maximum d'étiquettes attribuées par le SEE à un nœud.*

Les étiquettes d'un nœud sont les divers noms sous lesquels il est connu par les autres nœuds du réseau, et le processeur de communication dans chaque nœud doit pouvoir les reconnaître. Ces étiquettes constituent donc une partie de l'information nécessaire au routage qui doit être maintenue par le processeur de communication. Notre but étant la recherche de fonctions de routage nécessitant une information de routage compacte, seuls les k-SEE à valence indépendante du nombre de nœuds dans le réseau retiendront notre attention.

La notion de SEE permet de représenter l'information de routage de façon compacte. Pour démontrer qu'elle est effectivement utilisable, il nous reste à montrer comment générer des fonctions de routage sous forme de SEE valide, sans interblocage et à valence indépendante de la taille du réseau. C'est l'objet de la section suivante.

7.3 Méthodes de génération de SEE

On peut reprendre la problématique du routage par intervalles du chapitre 5 et la transposer dans le cas du routage par SEE. On se posera alors la question de savoir si une fonction de routage représentée par des tables de routage non compactes peut être représentée à l'aide d'un SEE (problème RI1 du chapitre 5). De même, on pourra considérer les problèmes analogues aux problèmes $RI2$ et $RI3$.

Pour les mêmes raisons qu'au chapitre 5, nous nous intéresserons aux problèmes de type $RI3$, c'est à dire à des méthodes qui permettent de générer une fonction de routage en même temps que sa représentation sous forme de SEE. Nous montrons d'abord que la notion de SEE contient le routage par intervalles classique, nous proposons ensuite une méthode générale de génération de SEE

7.3.1 SEE et routage par intervalles classique

La notion de SEE proposée contient la notion de schéma d'étiquetage par intervalles. Ceci est énoncé par la proposition 7.1 ci-dessous.

Proposition 7.1 *A tout schéma d'étiquetage par intervalles correspond un SEE de valence 1 qui induit la même fonction de routage.*

Démonstration : Soit $\mathcal{S}$, un schéma d'étiquetage par intervalles. Considérons le SEE $\mathcal{SE}$ construit à partir de $\mathcal{S}$ de la façon suivante :

- pour tout nœud x, $\mathcal{SE}_V(x) = \{\mathcal{S}_V(x)\}$
- pour tout nœud x et tout lien $l_1 \in L_x \bigcup \{\xi\}$, $l_2 \in L_x$, $\mathcal{SE}_E(x, l_1, l_2) = \mathcal{S}_E(l_2)$

Dans le schéma étendu $\mathcal{SE}$ tout nœud a donc une seule étiquette (la même que pour le schéma d'étiquetage par intervalles), et tous les liens d'un nœud ont le même jeu d'étiquettes (le jeu d'étiquettes attribué au nœud par $\mathcal{S}$). Il est immédiat que $\mathcal{SE}$ et S définissent la même fonction de routage.
$\odot$

La proposition 7.1 nous fournit donc une première méthode pour la génération de SEE, celle qui consiste à utiliser les solutions connues pour le routage par intervalles. Néanmoins une telle approche n'utilise pas toutes les possibilités offertes par un routeur basé sur la notion de SEE. Nous proposons ci-dessous une méthode générale de génération de fonctions de routage qui utilise mieux la puissance de représentation d'un SEE.

7.3.2 Une méthode de génération de SEE

Considérons l'anneau bidirectionnel de la figure 7.3. Pour construire une fonction de routage sans interblocage sur l'anneau, nous partons sur la même idée qu'au chapitre 4. Si on coupe la liaison entre les nœuds 0 et 6, on a un réseau linéaire pour lequel on connait un schéma d'étiquetage par intervalles sans interblocage. En effet, il suffit de garder les numéros des nœuds comme étiquettes des nœuds, et en chaque nœud i, d'associer l'étiquette $i+1$ au lien vers le nœud $i+1$, et l'étiquette 0 au lien vers le nœud $i-1$ (voir figure 7.3). La liaison entre 0 et 6 étant coupée, les liens de 0 vers 6 et de 6 vers 0 ne sont pas étiquetés.

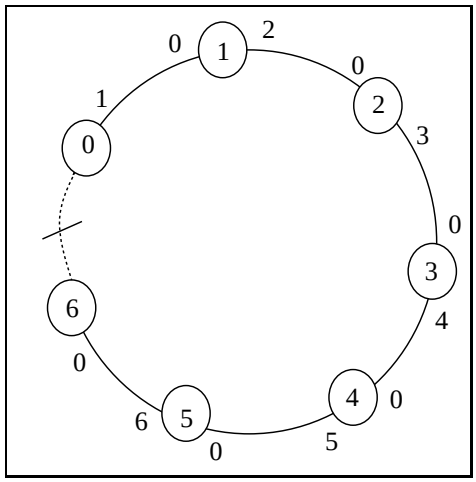

Figure 7.3: Schéma d'étiquetage par intervalles d'un anneau bidirectionnel

Nous montrons comment, à partir du schéma d'étiquetage par intervalles de l'anneau, construire un SEE valide et sans interblocage pour tout réseau admettant un cycle ou un chemin eulérien. Pour simplifier nous nous plaçons dans le cas ou un tel cycle ou chemin existe ; au besoin, comme au chapitre 4, chaque arête du réseau est remplacée par deux arêtes virtuelles pour avoir un graphe dont tous les sommets sont de degré pair.

Nous construisons d'abord un premier SEE valide et sans interblocage mais qui peut conduire à des chemins contenant des cycles. Nous montrons ensuite comment les cycles peuvent être supprimés en gardant les propriétés de validité et d'absence d'interblocage.

7.3.2.1 Un premier SEE qui peut donner des chemins avec cycles

Soit $G = (V, E)$ le graphe d'un réseau d'interconnexion. Rappelons que nous supposons que G admet un cycle eulérien. Considérons G' le graphe d'un cycle eulérien de G tel que défini au chapitre 4. D'après la définition de G', un nœud de G a plusieurs occurrences sur G' qui sont les sommets qui ont été créés lors de la visite du nœud dans le parcours du cycle eulérien. G' est un anneau bidirectionnel sur lequel nous pouvons appliquer le schéma d'étiquetage par intervalles introduit ci-dessus. Nous montrons comment le schéma d'étiquetage par intervalles sur G' peut être transformé en un SEE sur G. Nous commençons par introduire quelques notations.

Notation 7.3 *: Soit x un nœud de G, et $x_1 < x_2 < \ldots < x_k$ les occurrences de x sur G'. Nous noterons $l_i - (x)$ (resp. $l_i + (x)$) le lien de x_i à $x_i - 1$ (resp. $x_i + 1$)[1]. Dans la suite lorsqu'il ne peut y avoir de confusion sur le nœud x nous l'omettrons et utiliserons l_i- et l_i+ tout simplement.*

Ces notations sont illustrées par la figure 7.5 pour le réseau et le cycle eulérien de la figure 7.4.

Soit $\mathcal{SE}$ le SEE sur G défini comme suit :

- l'ensemble A des étiquettes est V' l'ensemble des sommets de G'
- pour tout nœud $x \in V$, l'ensemble des étiquettes attribuées à x est $\mathcal{SE}_V(x) = \{x' \in V' \mid x' \text{occurrence de } x\}$
- pour tout nœud $x \in V$,
  $\mathcal{SE}_E(x, l_i-, l_i+) = \mathcal{SE}_E(x, l_i+, l_i+) = \{x_i + 1\}$,
  $\mathcal{SE}_E(x, l_i-, l_i-) = \mathcal{SE}_E(x, l_i+, l_i-) = \{0\}$
- Pour tout nœud $x \in V$, le jeu d'étiquettes associé au lien interne ξ est le même que celui du lien définissant la plus petite[2] occurrence de x.

[1] Les opérations étant faites modulo $\mid V' \mid$.

[2] Ce choix est arbitraire, on peut choisir n'importe quelle autre occurrence

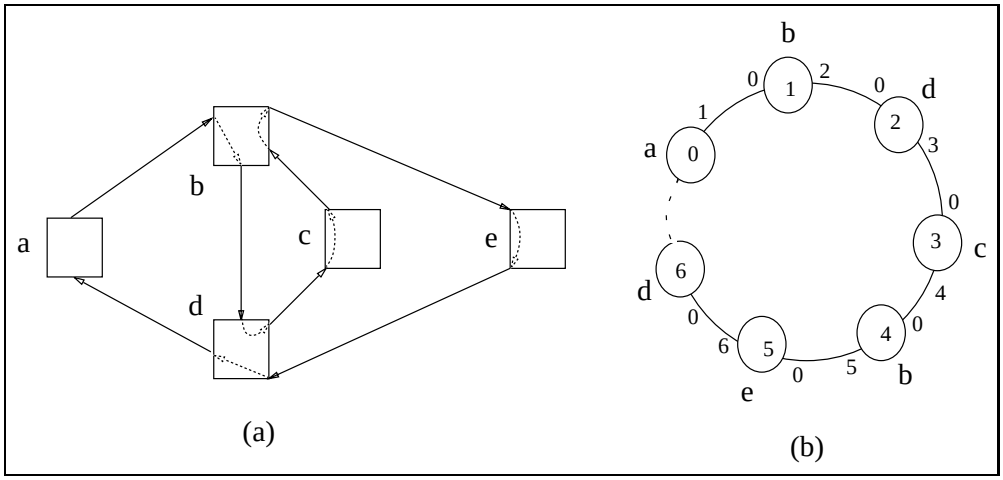

Figure 7.4: Un réseau G et le graphe G' d'un cycle eulérien associé

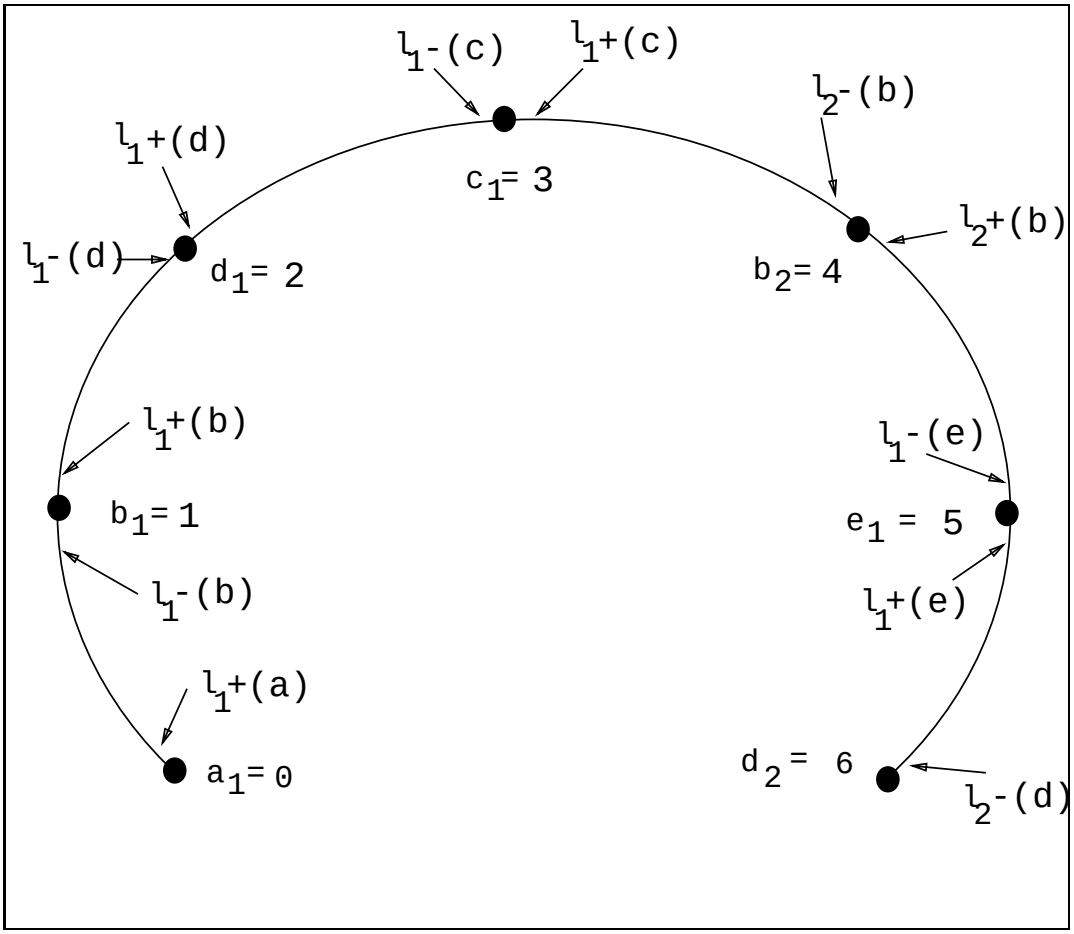

Figure 7.5: Illustration des notations (notation 7.3)

La figure 7.6 illustre $\mathcal{SE}$ sur le réseau et le cycle eulérien de la figure 7.4.

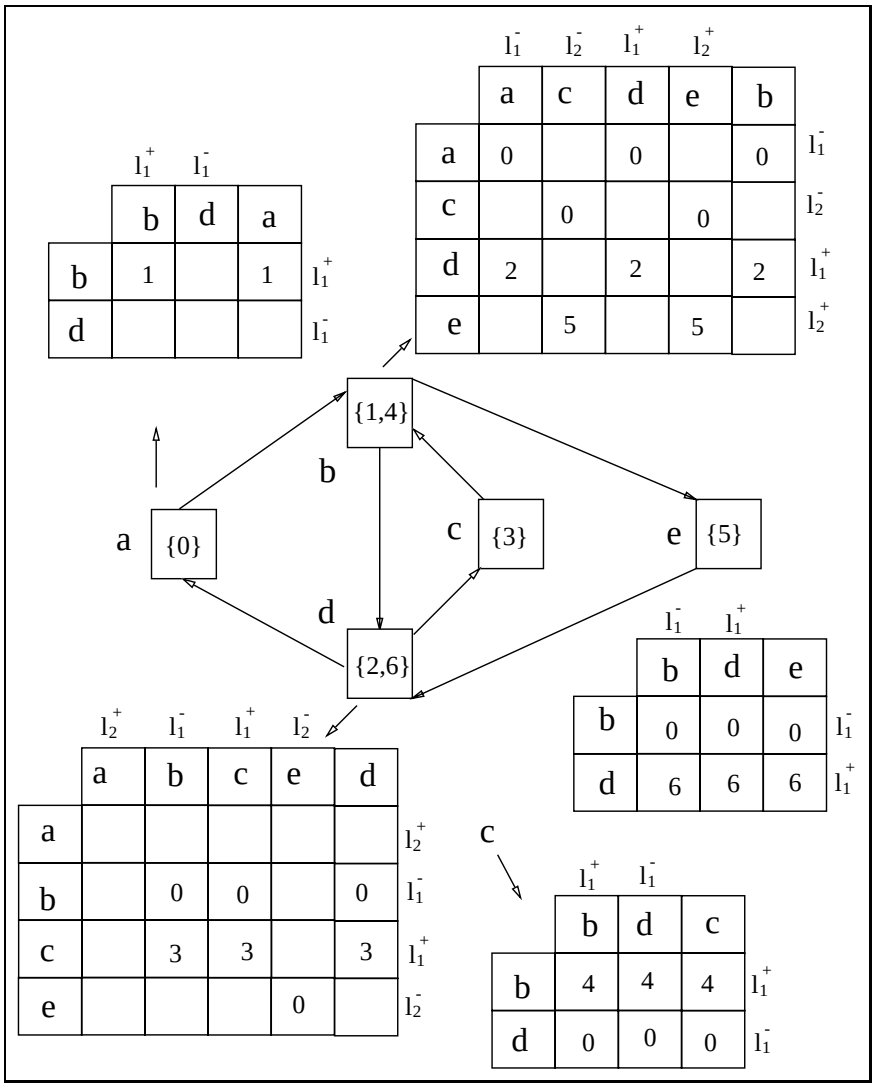

Figure 7.6: Un exemple de SEE $\mathcal{SE}$ (réseau de la figure 7.4)

Il est facile de voir que $\mathcal{SE}$ est simple et de valence $d/2$ où d est le degré maximum dans le réseau. Nous montrons maintenant qu'il est valide et sans interblocage.

Proposition 7.2 *$\mathcal{SE}$ est valide et sans interblocage.*

Démonstration : La proposition est évidente, en effet une fois que le paquet est introduit dans le réseau, le routage est rigoureusement identique au routage sur G' à la seule exception que quand un paquet arrive à destination il est délivré même si l'occurrence sur G' ne correspond pas à l'étiquette de destination dans l'entête du paquet.
⊙

Le SEE $\mathcal{SE}$ présente donc toutes les caractéristiques de correction. Cependant, un paquet peut passer plusieurs fois par un nœud intermédiaire avant d'arriver à destination. Dans le pire des cas, chaque nœud intermédiaire sera visité autant de fois qu'il a d'étiquettes. Par exemple, si on considère le SEE de la figure 7.6, un paquet émis par le nœud a à destination du nœud e décrira le chemin $abdcbe$ et passera donc 2 fois par le nœud b. Même si le nombre de fois qu'un paquet peut passer par un nœud est borné par le nombre d'étiquettes du nœud, les chemins sont inutilement rallongés.

Le problème avec $\mathcal{SE}$ est que dans chaque jeu d'étiquettes deux liens seulement sont étiquetés sans tenir compte du nombre de liens du nœud, et ceci a pour conséquence que tout chemin induit par $\mathcal{SE}$ est une sous chaîne du cycle eulérien. Nous montrons dans la section suivante comment, en étiquetant plus de deux liens dans un jeu d'étiquettes, dériver à partir de $\mathcal{SE}$ un SEE qui induit des chemins acycliques.

7.3.2.2 un SEE donnant des chemins sans cycles

Considérons un réseau d'interconnexion G pour lequel on a construit un SEE $\mathcal{SE}$ comme dans la section précédente.
Soit $\mathcal{ASE}$ le SEE défini de la façon suivante :

- les étiquettes attribuées aux nœuds sont les mêmes que ceux de $\mathcal{SE}$
- Les étiquettes associées aux liens en un nœud x sont données par les relations suivantes :

$$\mathcal{ASE}_E(x, l_i-, l_j-) = \{0\} \quad \text{si } j = i$$

$$\mathcal{ASE}_E(x, l_i-, l_j+) = \{x_j + 1\} \quad \text{si } x_j \neq Q-1 \text{ et } j \geq i$$

$$\mathcal{ASE}_E(x, l_i+, l_j-) = \begin{cases} \{x_{j-1}\} & \text{si } j \leq i \text{ et } j \neq 1 \\ \{0\} & \text{si } j = 1 \text{ et } x_1 \neq 0 \end{cases}$$

$$\mathcal{ASE}_E(x, l_i+, l_j+) = \{x_j + 1\} \quad \text{si } j = i \text{ et } x_j \neq Q-1$$

- en chaque nœud x le jeu d'étiquettes associé au lien interne ξ par $\mathcal{ASE}$ est le même que celui associé à ξ par $\mathcal{SE}$.

La figure 7.7 donne une illustration de $\mathcal{ASE}$ obtenu à partir du SEE de la figure 7.6. Les étiquettes encerclées dans les jeux d'étiquettes du nœud b correspondent aux liens étiquetés par $\mathcal{ASE}$ mais pas par $\mathcal{SE}$. L'ajout de ces nouvelles étiquettes permet de supprimer les cycles tout en conservant les propriétés de validité et d'absence d'interblocage. Avant d'énoncer et prouver ce résultat nous prouvons les trois lemmes suivants.

Lemme 7.1 *Soit s un nœud de G. Un paquet qui rentre en s par un lien $l_i+(s)$ et dont la destination est étiquetée d' avec $d' < s_i$, est retransmis sur le lien $l_j-(s)$ où s_j est la plus petite étiquette de s supérieure à d'.*

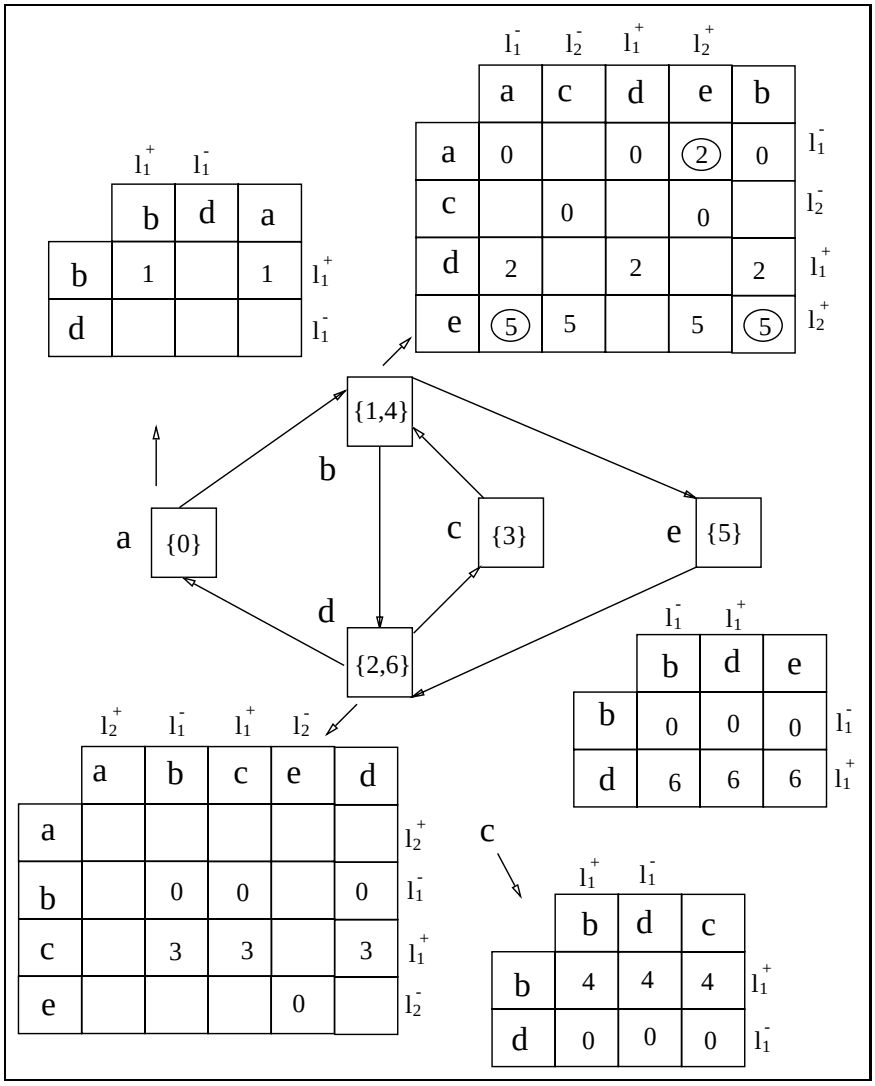

Figure 7.7: SEE acyclique

Démonstration : Il nous faut montrer que d' est dans l'intervalle associé à $l_j-(s)$ par le jeu d'étiquettes de $l_i+(s)$. Nous distinguons deux cas suivant la valeur de s_1.

Cas (1) : $s_1 \neq 0$.
Le jeu d'étiquettes associé à l_i+ est composé des singletons $\{0\}$, $\{s_1\}$, ..., $\{s_{i-1}\}$, $\{s_i\}$ associés respectivement aux liens l_1-, l_2-, ..., l_i-, et l_i+. Ce jeu d'étiquettes conduit aux intervalles $[0, s_1)$, $[s_1, s_2)$, ... $[s_{i-1}, s_i)$ et $[s_i, 0)$, et d' appartient à l'intervalle $[0, s_1)$ associé au lien l_1- si $j = 1$, ou à l'intervalle $[s_{j-1}, s_j)$ associé à l_j- si $j \neq 1$. Donc dans l'un ou l'autre cas d' appartient bien à l'intervalle associé au lien l_j-.

Cas (2) : $s_1 = 0$
Le jeu d'étiquettes associé à l_i+ est cette fois-ci composé des singletons $\{0 = s_1\}$, $\{s_2\}$, ..., $\{s_{i-1}\}$, $\{s_i\}$ associés respectivement aux liens l_2-, l_3-, ..., l_i-, et l_i+. Ce jeu d'étiquettes conduit aux intervalles $[0, s_2)$, $[s_2, s_3)$, ... $[s_{i-1}, s_i)$ et $[s_i, 0)$. Et donc d'après la définition de s_j, d' appartient à l'intervalle $[s_{j-1}, s_j)$ associé à l_j-.
$\odot$

Lemme 7.2 *Soit s un nœud de G. Un paquet arrivé en s par le lien $l_i-(s)$ et dont la destination est étiquetée d' avec $d' > s_i$ est retransmis par le lien $l_j+(s)$ où s_j est la plus grande étiquette de s inférieure à d'.*

Démonstration : Comme pour le lemme 7.1, il nous faut montrer que d' est dans l'intervalle associé à $l_j+(s)$ par le jeu d'étiquettes de $l_i-(s)$.
Appelons s_k la plus grande étiquette attribuée à s. Nous distinguons deux cas suivant que $s_k = Q-1$ ou $s_k \neq Q-1$.

Cas (1) : $s_k \neq Q-1$
Le jeu d'étiquettes associé à l_i- est $\{0\}$, $\{s_i+1\}$, ..., $\{s_k+1\}$ correspondants respectivement aux liens l_i-, l_i+, $l_{i+1}+$, ..., l_k+. Ce jeu d'étiquettes conduit aux intervalles $[0, s_i+1)$, $[s_i+1, s_{i+1}+1)$, ..., $[s_{k-1}+1, s_k+1)$ et $[s_k+1, 0)$. d' étant supérieur à s_i, d' est dans l'intervalle $[s_j+1, s_{j+1}+1)$ si $s_j \neq s_k$, et dans l'intervalle $[s_k+1, 0)$ si $s_j = s_k$.

Cas (2) : $s_k = Q-1$
Le jeu d'étiquettes associé à l_i- est cette fois-ci $\{0\}$, $\{s_i+1\}$, ..., $\{s_{k-1}+1\}$. Ce jeu d'étiquettes induit les intervalles $[0, s_i+1)$, $[s_i+1, s_{i+1}+1)$, ..., $[s_{k-1}+1, 0)$, et comme au cas (1), on montre que d' est dans le bon intervalle c'est à dire $[s_j+1, s_{j+1}+1)$ si $s_j \neq s_{k-1}$, ou $[s_{k-1}+1, 0)$ si $s_j = s_{k-1}$.
$\odot$

Proposition 7.3 *Le routage défini par $\mathcal{ASE}$ induit des chemins sans cycles.*

Démonstration : La proposition est une conséquence des lemmes 7.1 et 7.2. En effet, d'après ces deux lemmes, le routage induit par $\mathcal{A}SE$ utilise soit un sens du cycle eulérien, soit l'autre; et à chaque fois il n'y a pas d'autres occurrences du nœud courant entre le voisin choisi pour retransmettre le paquet et la destination.
$\odot$

Proposition 7.4 *$\mathcal{A}SE$ est sans interblocage.*

Démonstration : $\mathcal{S}E$ étant sans interblocage, pour montrer que $\mathcal{A}SE$ est sans interblocage il nous suffit de montrer que les chemins induits par $\mathcal{A}SE$ sont des sous-chemins de ceux induits par $\mathcal{S}E$ et que les arcs sont empruntés dans le même ordre.
Etant donnés les lemmes 7.1 et 7.2, il suffit de montrer qu'au départ le paquet est envoyé dans la bonne direction. Ceci est garanti par le fait qu'en tout nœud le jeu d'étiquettes associé au lien interne par les deux schémas $\mathcal{S}E$ et $\mathcal{A}SE$ est le même.
$\odot$

Proposition 7.5 *$\mathcal{A}SE$ est valide*

Démonstration : Il nous faut montrer que tout paquet émis par un nœud s vers un nœud d arrive à destination au bout d'un nombre fini de pas. Comme d'après la procédure de routage chacune des étiquettes de la destination peut être utilisée dans l'entête d'un paquet, cela revient à montrer que toute étiquette de la destination conduit à un chemin de longueur fini.
Soit d' une étiquette de la destination. Nous distinguons les deux cas $d' < s_1$ et $d' > s_1$.

- Si $d' < s_1$, s transmet le paquet sur $l_1 - (s)$. Soit on arrive à destination auquel cas le paquet est délivré, soit on arrive en un nœud u par un lien $l_i + (u)$ avec $u_i > d'$ et, d'après le lemme 7.1, le paquet arrivera à destination.
- Si $d' > s_1$, considérons s_j la plus grande étiquette de s qui est inférieure à d'. D'après le jeu d'étiquettes associé au lien interne de s, s retransmettra le paquet sur $l_j + (s)$. On arrive alors en un nœud v par le lien $l_r - (v)$ avec $v_r \leq d'$. Si v est une des étiquettes de la destination le paquet est délivré, dans le cas contraire le lemme 7.2 nous garantit que le paquet arrivera à destination au bout d'un nombre fini de pas.

$\odot$

7.3.3 Qualité du SEE $\mathcal{A}SE$

Pour évaluer le SEE proposé nous avons calculé les mêmes paramètres qu'aux chapitres 4 et 6. Pour calculer le meilleur chemin entre un nœud source et un

nœud destination, nous avons simulé le routage en utilisant toutes les étiquettes de la destination et nous avons retenu celle qui donne le chemin le plus court. Les résultats sont donnés par les tables de la figure 7.8. Nous avons utilisé les mêmes cycles eulériens qu'au chapitre 4.

(n,k)	R1	R2
(2,4)	1.5	2
(2,5)	2	4
(2,6)	1.67	4.33
(2,7)	2	6,33
(2,8)	1,75	6.5
(2,9)	2	8.50
(2,10)	1.8	8.60
(3,4)	1.5	1.67
(3,5)	2	2.17
(3,6)	1.67	2.11
(3,7)	2	2.44

(a)

(n,k)	R1	R2
(2,4)	3	3
(2,5)	4	4
(2,6)	5	6.50
(2,7)	6	9.50
(2,8)	7	13
(2,9)	8	17
(2,10)	9	21.50
(3,4)	3	4
(3,5)	4	5
(3,6)	5	6
(3,7)	6	7

(b)

(n,k)	R1	R2
(2,4)	1.17	1.21
(2,5)	1.27	1.37
(2,6)	1.29	1.43
(2,7)	1.34	1.50
(2,8)	1.35	1.52
(2,9)	1.39	1.55
(2,10)	1.40	1.56
(3,4)	1.25	1.29
(3,5)	1.35	1.44
(3,6)	1.37	1.47
(3,7)	1.43	1.53

(c)

Figure 7.8: Qualité du SEE acyclique : (a) : rapport $D_{R,G}/D_G$, (b) : $\rho_{R,G}$, (c) : $\overline{\rho_{R,G}}$

Ces caractéristiques montrent que le SEE donne des chemins de meilleure qualité que les schémas d'étiquetage par intervalles du chapitre 6. Néanmoins, comme l'on pouvait s'y attendre, la qualité des chemins fournis par le SEE est moins bonne que celle des chemins donnés par la fonction de routage présentée au chapitre 4 qui, elle aussi, est basée sur un cycle eulérien mais utilise une information de routage non compacte. Ceci peut s'expliquer par le fait que l'utilisation de tables de routage classiques permet de représenter plus de fonctions que l'utilisation d'une information de routage compacte. Avec des tables de routage non compactes on peut, notamment dans le cas ou il y a plusieurs chemins, faire un calcul préalable pour ne garder que les meilleurs chemins. En général on ne pourra pas toujours trouver un SEE pour représenter le resultat d'un tel calcul. La notion de SEE adaptatif que nous introduisons ci-dessous permet de garder

les diverses possibilités en laissant la décision à l'exécution.

7.4 SEE adaptatif

Jusqu'à maintenant les techniques de représentation compacte de l'information de routage que nous avons examinées ne permettent de représenter que des fonctions de routage déterministes. Ceci est dû au fait que, pour le schéma d'étiquetage classique et le SEE, les intervalles correspondant à un jeu d'étiquettes sont disjoints deux à deux. Dans cette section, nous proposons une extension de la notion de SEE qui, pour prendre en compte le routage adaptatif, utilise des intervalles non forcément disjoints.

7.4.1 Définitions

Définition 7.9 (jeu d'intervalles) *: Soit A un ensemble d'étiquettes. Un jeu d'intervalles associé à un nœud x est la donnée d'une application I de L_x dans $\mathcal{P}(A \times A)$ qui, à un lien $l \in L_x$ associe un ensemble $I(l)$ de couples d'étiquettes. Chaque couple (a, b) correspondant aux deux bornes de l'intervalle $[a, b)$.*

Contrairement à un jeu d'étiquettes dans lequel les intervalles induits sont disjoints deux à deux, les intervalles d'un jeu d'intervalles peuvent se recouvrir.

Définition 7.10 *: Un jeu d'intervalles est dit déterministe si les intervalles associés à un nœud sont disjoints deux à deux. Dans le cas contraire, le jeu d'intervalles est dit adaptatif.*

La notion de jeu d'intervalles adaptatif nous permet d'étendre la notion de SEE pour prendre en compte le routage adaptatif.

Définition 7.11 (SEE adaptatif) *: Un SEE adaptatif est la donnée de*

- *une numérotation multiple des nœuds de G, et*
- *pour chaque nœud x, $\mid L_x \cup \{\xi\} \mid$ jeux d'intervalles adaptatifs relatifs à x associés chacun à un lien $l \in L_x$.*

Un SEE adaptatif induit un routage adaptatif de façon suivante. En chaque nœud, on choisit un lien parmi ceux dont les intervalles associés contiennent l'étiquette de la destination.

Un SEE adaptatif répond au critère de compactage de l'information de routage. Les intervalles se recouvrant, il est seulement nécessaire de représenter les deux bornes.

La notion de SEE adaptatif contient celle de SEE. En effet, un SEE est un SEE adaptatif dans lequel les intervalles d'un jeu d'intervalles sont disjoints deux à

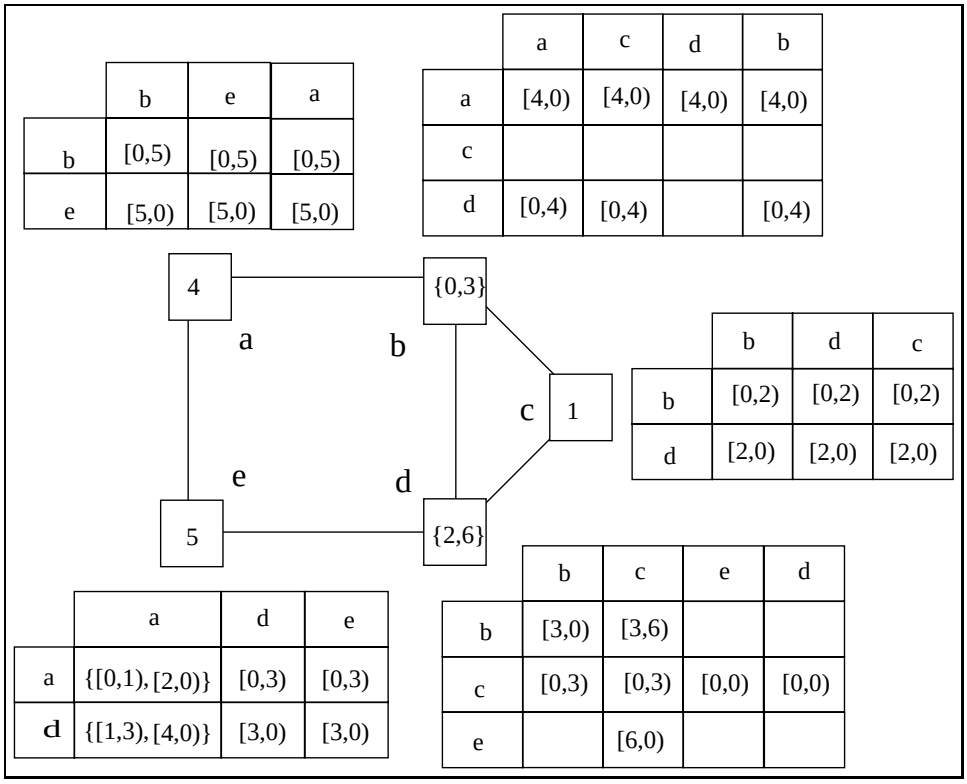

Figure 7.9: Un exemple de SEE adaptatif

deux et forment donc une partition de A ; ils ne sont alors représentés que par leurs bornes inférieures qui servent comme étiquettes des liens.
La figure 7.9 montre un exemple de SEE adaptatif. Pour illustrer la procédure de routage, reprenons l'exemple de la section précédente dans lequel le nœud a veut envoyer un message au nœud d et choisit 6 comme étiquette de destination. Le routage est effectué de la manière suivante : a consulte le jeu d'intervalles associé au lien interne ξ et trouve que l'étiquette 6 est dans l'intervalle $[5, 0)$ associé au lien vers e. a retransmet donc le paquet vers e. Au nœud e, le jeu d'intervalles associé au lien d'entrée du paquet (colonne a) est adaptatif car un des deux intervalles associés au lien vers a ($[2, 0)$) a une intersection non vide avec les deux intervalles $[1, 3)$ et $[4, 0)$ associés au lien vers d. Notamment l'étiquette 6 est à la fois dans l'intervalle $[2, 0)$ et $[4, 0)$; e a donc le choix soit de renvoyer le paquet vers a, soit de le retransmettre vers d.

Comme pour les SEE, un SEE adaptatif ne définit pas forcément une fonction de routage correcte. Pour caractériser les fonctions de routage représentées par un SEE adaptatif les propriétés de validité, optimalité, absence d'interblocage, et conservation de voisinage sont définies de la même manière que pour le SEE. De même, pour caractériser la taille de l'information de routage, les notions de SEE adaptatif simple, k-SEE adaptatif, et de valence d'un SEE adaptatif sont naturellement déduites de celles concernant le SEE.

7.4.2 Méthodes de génération de SEE adaptatifs

La notion de SEE adaptatif contient toutes les techniques de représentation compacte de l'information de routage que nous avons examinées jusqu'à présent. Toutes les méthodes de génération de schémas d'étiquetage par intervalles et de SEE que nous avons proposées peuvent donc être utilisées dans un fonctionnement "dégradé " d'un routeur basé sur le SEE adaptatif. Dans cette section, nous montrons comment la méthode de génération de SEE de la section précédente peut être étendue pour générer des SEE adaptatifs.

Considérons un réseau d'interconnexion G sur lequel on a construit un SEE $\mathcal{SE}$ comme dans le section 7.3.2.1. Soit $\mathcal{DSE}$ le SEE adaptatif défini de la façon suivante :

- les étiquettes attribuées aux nœuds sont les mêmes que ceux de $\mathcal{SE}$
- Soit x un nœud de G, et $x_1, \ldots, x_k$ ses étiquettes. Nous prendrons comme convention $x_{k+1} = 0$ et $x_0 = Q - 1$, et les opérations sur les x_i sont faites modulo Q.
  Les intervalles associés aux liens du nœud x sont donnés par les relations suivantes :

$$\mathcal{DSE}_E(x, l_i-, l_j-) = \begin{cases} \{[0, x_i)\} & \text{si } j = i \\ \{[x_{j-1} + 1, x_j)\} & \text{si } j > i \end{cases}$$

$$\mathcal{DSE}_E(x, l_i-, l_j+) = \{[x_j + 1, x_{j+1}\} \quad \text{si } x_j \neq Q - 1 \text{ et } j \geq i$$

$$\mathcal{DSE}_E(x, l_i+, l_j-) = \begin{cases} \{[x_{j-1} + 1, x_j)\} & \text{si } j \leq i \text{ et } j \neq 1 \\ \{[0, x_1)\} & \text{si } j = 1 \text{ et } x_1 \neq 0 \end{cases}$$

$$\mathcal{DSE}_E(x, l_i+, l_j+) = \{[x_j + 1, 0)\} \quad \text{si } j = i \text{ et } x_j \neq Q - 1$$

- en chaque nœud x le jeu d'étiquettes associé au lien interne ξ par $\mathcal{DSE}$ est le même que celui associé à ξ par $\mathcal{SE}$.

Le SEE adaptatif $\mathcal{DSE}$ correspond donc au SEE déterministe $\mathcal{ASE}$ auquel on a rajouté certains intervalles pour autoriser, en plus des chemins induits par $\mathcal{ASE}$, des chemins qui sont composés d'une suite d'arcs directs suivie d'une suite d'arcs indirects[3]. Ceci est fait en autorisant un paquet arrivé au nœud x par le lien l_i- (arc direct) et dont l'étiquette de destination est comprise entre les étiquettes x_j et x_{j+1} d'emprunter au choix l'un des deux liens l_j+ et $l_{j+1}-$. Il est immédiat que $\mathcal{DSE}$ est valide et sans interblocage.
La figure 7.10 illustre la méthode sur le réseau qui nous a servi d'exemple tout au long de ce chapitre. Les intervalles encerclés correspondent aux intervalles qui ont été rajoutés par rapport à $\mathcal{ASE}$.

[3]Voir chapitre 4 pour les définitions

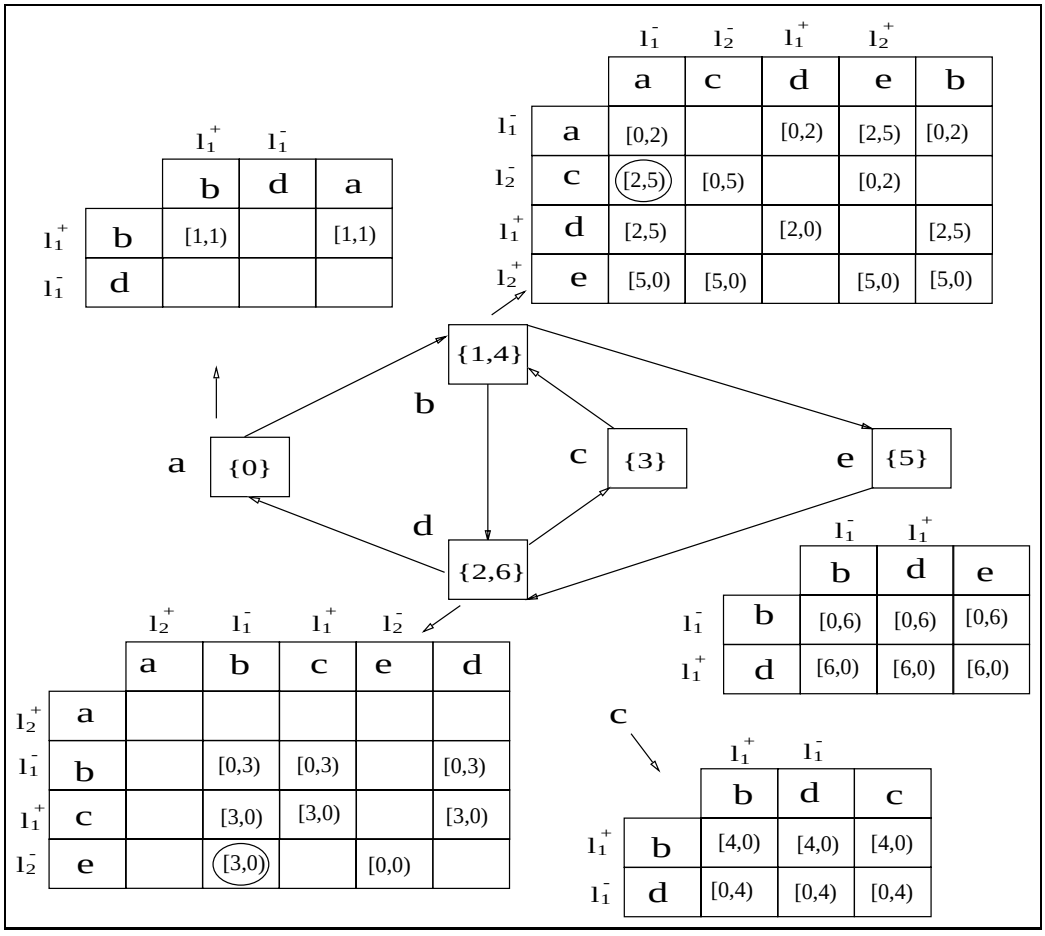

Figure 7.10: SEE adaptatif $\mathcal{D}SE$ pour le réseau de la figure 7.4

7.5 Conclusion

Dans ce chapitre nous avons proposé une nouvelle technique de représentation compacte de l'information de routage (le SEE) et avons présenté des méthodes générales de génération de tables de routage pour cette technique. Les évaluations de notre méthode de génération de SEE montrent un certain gain par rapport aux méthodes de génération de schémas d'étiquetage par intervalles classiques étudiées aux chapitres 5 et 6. Pour prendre en compte le routage adaptatif, nous avons défini la notion de SEE adaptatif et montré comment on peut générer des SEE adaptatifs. Néanmoins des investigations plus poussées sont à faire pour trouver des schémas qui utilisent mieux la puissance de représentation des SEE.

Partie III

Conclusion

Chapitre 8

Conclusion et perspectives

Conclusion : Bilan de la contribution sur le routage

L'objectif de ce travail était l'étude du contrôle des communications dans les machines massivement parallèles. Après avoir montré à travers un exemple les limites de l'utilisation de la reconfiguration dynamique pour la communication dans les machines parallèles de grande taille, nous avons considéré l'alternative du routage. Nous nous sommes intéressés aux techniques de routage sans interblocage en nous attachant à une caractéristique essentielle des machines massivement parallèles, l'extensibilité.

Pour la génération actuelle des machines parallèles disposant de peu de support matériel pour la communication, nous avons proposé une nouvelle méthode de génération de fonctions de routage sans interblocage qui, contrairement aux techniques classiques de prévention de l'interblocage basées sur l'ordonnancement des tampons, nécessite un nombre de tampons par processeur indépendant de la taille de la machine. Notre méthode, basée sur l'interdiction de certaines dépendances, est suffisamment générale et permet de générer aussi bien des fonctions de routage adaptatives que déterministes. Les évaluations faites de la fonction de routage déterministe utilisant les chemins les plus courts montrent que les chemins fournis sont proches de l'optimum. Enfin, signalons que notre méthode a été implantée dans le noyau de communication de ParX ainsi que dans MARC, un outil d'aide au placement pour réseaux de transputers dévéloppé à l'université de Berne [Boillat91].

La deuxième partie de notre étude a porté sur les techniques de représentation compacte de l'information de routage. Nous avons d'abord considéré la technique du routage par intervalles qui utilise une table de routage par processeur dont la taille est de l'ordre du nombre de voisins. Le schéma d'étiquetage par intervalles $\mathcal{T}$ que nous avons proposé pour les tores permet d'utiliser les liens de rebouclage

sans introduire l'interblocage et donne des chemins proches de l'optimum.
Pour les réseau généraux, la méthode d'étiquetage $\mathcal{G}S_1$ améliore le routage sur un arbre recouvrant de Santoro et Khatib en utilisant tous les liens au moins dans un sens.

Partant de la constatation que la classe des fonctions qui peuvent être représentées par des tables de routage par intervalles est plus réduite que celle des fonctions décrites par des tables de routage classiques, nous avons étendu la notion de schéma d'étiquetage par intervalles et proposé le SEE. Le SEE utilise des tables de routage par processeur de communication dont la taille est de l'ordre du carré du nombre de voisins, et peut représenter des fonctions de routage adaptatives. L'évaluation de la méthode de génération de SEE proposée montre un gain notable par rapport aux autres méthodes de routage par intervalles étudiées.

Perspectives

Dans le domaine de l'architecture des machines parallèles, la tendance actuelle est l'intégration dans le matériel de la plupart des fonctions de communication. Les perspectives de notre travail porterons donc essentiellement sur les méthodes de routage utilisant une représentation compacte de l'information de routage.

Peu de travaux ont porté sur la génération de tables de routage par intervalles. Nous pensons que ce point mérite de continuer à être étudié surtout avec l'arrivée prochaine sur le marché d'un processeur de communication, le C104 d'INMOS, utilisant le routage par intervalles. Les recherches peuvent porter notamment sur les schémas d'étiquetage par intervalles qui répartissent "le plus équitablement possible" les chemins entre les liens, et sur le routage adaptatif en utilisant des intervalles recouvrants.

La technique de routage par SEE que nous avons proposée est suffisamment simple pour être intégrée dans le matériel, et être donc à la base d'un processeur de communication. Le travail sur ce point doit être poursuivi pour trouver de nouvelles méthodes de génération de SEE notamment pour les réseaux réguliers les plus couramment utilisés pour la construction des machines parallèles.

Le travail présenté dans ce rapport ne concerne que la communication point à point impliquant uniquement deux processeurs en même temps. Beaucoup de modèles de programmation nécessitent des schémas de communication plus complexes faisant intervenir plusieurs processeurs. Il serait intéressant d'étudier la mise en œuvre de ces schémas en recherchant des solutions qui puissent être efficacement intégrées dans le matériel.

Enfin, en revenant au problème plus général de la programmation des machines parallèles, parmi les trois points que nous avons évoqués dans l'introduction, on peut considérer que les études sur les mécanismes de base pour la communication sont les plus avancées. Une large utilisation, non ésotérique, des machines parallèles nécessitera un effort important sur des techniques d'extraction automatique du parallélisme et les méthodes de placement des tâches sur les processeurs.

Sommaire

1 INTRODUCTION **1**
1.1 Du séquentiel au parallèle 1
1.2 Les architectures parallèles 2
1.3 Programmation et contrôle des machines parallèles 3
1.3.1 Le problème de partitionnement 5
1.3.2 L'allocation - l'ordonnancement 6
1.3.3 Contrôle des communications et synchronisations 7
1.4 Contribution de cette thèse 10
1.5 Organisation de la suite du rapport 11

I Un mécanisme de base pour la communication : l'acheminement des messages **13**

2 Acheminement des messages par reconfiguration dynamique **15**
2.1 Introduction 15
2.2 Architectures des supernodes 17
2.2.1 Réseaux de Clos à 3 étages 17
2.2.2 Dispositif de commutation des supernodes 18
2.3 Algorithmes de configuration des supernodes 20
2.3.1 Définitions 20
2.3.2 Modélisation du contrôle 21
2.3.3 Algorithmes de configuration 23
2.4 Acheminement de messages par reconfiguration 28

3 Acheminement des messages par routage **29**
3.1 Introduction 29
3.2 Organisation et fonctionnement d'un routeur 29
3.3 Techniques d'acheminement 31
3.3.1 La commutation de circuits 31
3.3.2 La commutation de paquets et de messages 32
3.3.3 Sur la mise en œuvre des techniques 34
3.4 Les principaux problèmes du routage 34
3.4.1 Abscence de famine et équité 34

3.4.2 interblocage . . . 35
3.4.3 Validité . . . 36
3.5 La fonction de routage . . . 37
3.5.1 Les modes de routage . . . 37
3.5.2 Représentation de la fonction de routage . . . 37
3.6 Méthodes de résolution du problème d'interblocage . . . 38
3.6.1 Techniques de détection-guérison . . . 38
3.7 Techniques de prévention de l'interblocage . . . 39
3.7.1 Méthode nécessitant un contrôle à l'exécution . . . 39
3.7.2 Méthodes basées sur les graphes de tampons . . . 42
3.8 Prévention de l'interblocage avec un nombre constant de classes de tampons . . . 48
3.8.1 Les solutions dédiées . . . 48
3.8.2 Les solutions générales . . . 48

4 Une nouvelle méthode de routage sans interblocage utilisant un nombre constant de tampons 51
4.1 Introduction . . . 51
4.2 Notations et définitions . . . 52
4.3 Routage sans interblocage sur un anneau bidirectionnel . . . 55
4.4 Une nouvelle classe de méthodes de routage sans interblocage . . . 58
4.4.1 Une formulation du problème de routage sans interblocage 58
4.4.2 Méthode basée sur un arbre recouvrant . . . 60
4.5 Méthode du cycle eulérien . . . 60
4.5.1 Principe de la solution . . . 60
4.5.2 Correction de la méthode . . . 61
4.5.3 Evaluation de la méthode . . . 62
4.5.4 Application : une fonction de routage déterministe . . . 63
4.6 Conclusion . . . 67

II Compactage de l'information de routage 69

5 Introduction et état de l'art 73
5.1 Motivations . . . 73
5.2 Méthodes de compactage de l'information de routage . . . 75
5.2.1 Routage hiérarchique . . . 75
5.2.2 Routage par préfixes (Prefix-Routing) . . . 77
5.2.3 Routage par intervalles . . . 77
5.2.4 Problématique du routage par intervalles . . . 78
5.2.5 Définitions et notations . . . 80
5.2.6 Caractérisation des schémas d'étiquetage . . . 82
5.3 Méthodes de routage par intervalles . . . 83
5.3.1 Une famille de réseaux réguliers : grilles et hypercubes . . . 83

5.3.2 Etiquetage pour réseaux généraux 84
5.4 Conclusion : bilan de l'état de l'art 88

6 Nouveaux schémas d'étiquetage par intervalles 89
6.1 Un réseau régulier très utilisé : le tore 89
6.1.1 Schémas d'étiquetage 89
6.1.2 Un nouveau schéma d'étiquetage pour le tore 90
6.1.3 Preuve de correction du schéma T 90
6.1.4 Qualité des chemins induits par T 96
6.2 Une nouvelle méthode d'étiquetage pour réseaux généraux 96
6.2.1 Etiquetage des nœuds 97
6.2.2 Etiquetage des liens 97
6.2.3 Preuve de correction 98
6.2.4 Qualité des schémas GS_1 101
6.3 Conclusion 102

7 Une nouvelle technique de représentation compacte de l'information de routage : le schéma d'étiquetage étendu (SEE) 103
7.1 Introduction 103
7.2 Schéma d'étiquetage étendu (SEE) 104
7.2.1 Définitions et notations 104
7.2.2 exemple 105
7.2.3 Interprétation d'un SEE en termes de routage 106
7.2.4 Caractérisation d'un SEE 107
7.3 Méthodes de génération de SEE 108
7.3.1 SEE et routage par intervalles classique 108
7.3.2 Une méthode de génération de SEE 109
7.3.3 Qualité du SEE ASE 116
7.4 SEE adaptatif 118
7.4.1 Définitions 118
7.4.2 Méthodes de génération de SEE adaptatifs 120
7.5 Conclusion 122

III Conclusion 123

8 Conclusion et perspectives 125

Bibliographie

[Agrawal86] D.P. Agrawal, V.K. Janakiran. *Evaluating the performance of multicomputer configurations.* IEEE Compute, May 1986, pp. 23-37.

[Ahuja79a] V. Ahuja. *Algorithm to check network states for deadlock.* IBM J. Res. develop. vol. 23 no. 1, 1979.

[Ahuja79b] V. Ahuja. *Routing and flow control in Systems Network Architecture.* IBM System Journal, Vol. 18, no. 2, 1979.

[Alhafez91] N. Alhafez. *Etude et mise en œuvre de routages non blocants dans les architectures multi-processeurs.* Thèse Université Claude Bernard - Lyon 1, Décembre 1991.

[Allen88] F. Allen, M. Burke, P. Charles, R. Cytron, J. Ferrante. *An overview of the PTRAN analysis system for multiprocessing.* Journal of Parallel and Distributed Computing 5 (617-640) 1988.

[Anderson84] R. Anderson, E. Mayr. *Parallelism and Greddy algorithms.* Technical Report STAN-CS-84-1003, Dept. of Computer Science, Stanford University, April 1984.

[Andresen77] S. Andresen. *The looping algorithm extended to base 2^t rearrangeable switching networks.* IEEE Trans. on Computers, Vol c-25, 1977, pp. 1057-1063.

[Anger90] F.D. Anger, J.J. Hwang and Y.-C. Chow. *Scheduling with sufficient Loosely Coupled Processors.* Journal of Parallel and Distributed Computing 9, 87-92 (1990).

[Annot87] J. K. Annot and R. A. H. Van Twist. *A novel deadlock free and starvation free packet switching communication processor.* Lectures and Notes Computer Science no 258 juin 1987.

[Antonelli89] S. Antonelli, F. Baiardi, S. Pelagatti, M. Vanneschi. *Communication Cost and Process Mapping in Massively Parallel*

Systems: a Static Approach. Technical Report : TR-12/89 Universta Degli Studi di Pisa, Dipartimento di Informatica

[Awerbuch90] B. Awerbuch, A. Bar-Noy, N. Linial and D. Peleg. *Improved Routing Strategies with Succint Tables.* Journal of Algorithms 11, 307-341 (1990).

[Bal89] H.E. Bal, J.G. Steiner, A.S. Tanembaum. *Programming Languages for Distributed Computing Systems.* ACM Computing Surveys, vol. 21, no. 3, September 1989.

[Balaniuk93] A. Balaniuk, H. Castro, R. Despons, F. Menneteau, L. Mugwaneza, T. Muntean, E-G. Talbi, P. Waille. *Paros : A generic multi virtual machines parallel operating system* Proc. PARCO'93, Grenoble (France), September 7-10, 1993.

[Bakker90] E.M. Bakker, J. van Leeuwen, R.B. Tan. *Prefix routing schemes in dynamic networks.* Tech. Rep. RUU-CS-90-10, Dept. of Computer Science, Utrecht University, March 1990

[Bakker91] E.M. Bakker, J. van Leeuwen, R.B. Tan. *Linear Interval Routing.* Algorithm Review, vol. 2, no. 2, 1991.

[Bhatkar90] V.P. Bhatkar. *Parallel computing : An indian perspective.* Proc. of CONPAR 90 - VAPP IV, LNCS 457, Springer-Verlag H. Burkhart (Ed), 1990.

[Beeten87] J. Beetem, M. Denneau and D. Weingarten. *The GF11 Parallel Computer.* Experimental Parallel Computing Architectures, North Holland, J. Dongarra Ed. North Holland, Amsterdam 1987.

[Berge83] C. Berge. *Théorie des graphes.* Gauthier-Villars 1983.

[Blazewicz87] J. Blazewics, J. Brzezinski, G. Gambosi. *Time-Stamp Approach to store-and-forward deadlock prevention.* IEEE T. COM. vol. COM-35, no. 5, May 1987.

[Blume92] W.Blume, R. Eigenman. *Performance analysis of parallelizing compilers on the Perfect Benchmarks programs.* IEEE Trans. on Parallel and Distributed Systems, vol 3 no. 6, November 1992

[Boillat91] J. E. Boillat, N. Iselin and P. G. Kropf. *MARC : A Tool for Automatic Configuration of Parallel Programs.* Transputing '91, P. Welch et al. Eds. IOS press, 1991.

[Bouloutas89] A. Bouloutas and P.M. Gopal. *Some Graph Partitioning Problems and Algorithms Related to Routing in Large Computer Networks.* Proc. of the 9th Int. Conf. on Distributed Computing, Newport Beach, Calfornia 1989.

[Castro93] H. Castro, A. Elleuch, T. Muntean and P. Waille. *Generic Microkernel Architecture for the PAROS PARallel Operating System* Proc. Transputer and Applications Systems'93, R. Grebe et al. (Eds.) IOS Press, 1993.

[Chan87] C. W. Chan and T. S. P. Yum. *An algorithm for detecting and resolving Store and Forward deadlocks in Packet-Switching networks.* IEEE Transactions on Communications Vol com 35 no 8 1987.

[Chandy79] K.M. Chandy and J. Misra. *Deadlock Abscence Proofs for Networks of Communicating Processes.* Information Processing Letters, Volume 9, number 4, 1979.

[Chang90] L.C. Chang, B.T. Smith. *Classification and evaluation of parallel programming tools.* T.R. no. CS90-22, Dept. of Computer Science, The University of New Mexico, Albuquerque, November 1990.

[Cidon87] I. Cidon, J.M. Jaffe and M. Sidi. *Distributed Store and Forward deadlock detection and resolution algorithm.* IEEE Transactions on Communications Vol. com. 35 no 11 1987.

[Clos53] C. Clos. *A study of non blocking switching networks.* The bell system technical journal, March 1953.

[Cytron89] R. Cytron, M. Hind, W. Hsieh. *Automatic generation of DAG parallelism.* ACM SIGLPAN, vol. 54, no 6, 1989

[Dally86] W. J. Dally and C. L. Seitz. *The torus routing chip.* Distributed Computing 1986.

[Dally87] W. J. Dally and C. L. Seitz. *Deadlock-free message routing in multiprocessor interconnection networks.* IEEE Transactions on Computers Vol c36 no 5 1987.

[Dolter88] J.W. Dolter, P. Ramanathan and K.G. Shin. *A VLSI Architecture for Dynamic Routing in HARTS.* CRL-TR-04-88, The University of Michigan Computing Research Laboratory.

[Dolter89] J.W. Dolter, R. Ramanathan, K.G. Shin. *A Microprogramable VLSI Routing Controller for Haartz.* Proc. IEEE Int. Conf. on Computer VLSI Design, October 1989.

[Dolter91] J.W. Dolter, R. Ramanathan, K.G. Shin. *Performance Analysis of Virtual Cut-through switching in Haartz: A Hexagonal Mesh Multicomputer.* IEEE T. Comp. vol. 40, no. 6, June 1991.

[Eudes87] J. Eudes, F. Menneteau, L. Mugwaneza, T. Muntean. *PDS : Advanced Program Development System for transputer based machines.* Proc. of the 10th OCCAM User Group, A. Bakkers(ed.) (15-28) , IOS press 1989.

[Feng81] T.Y. Feng. *A survey of interconnection networks.* IEEE Computer, December 1981.

[Foster90] I. Foster, S. Taylor. *STRAND : new concepts in parallel programming.* Prentice hall, Englewood Cliffs, NJ, 1990

[Gabow76] H. N. Gabow. *Using Euler Partitions to edge color bipartite multigraph.* International Journal of Computer and Information Sciences, Vol 5 no 4, 1976, pp 345-355.

[Garey79] M.R. Garey, D.S. Jonhson. *Computers and Intractability : A guide to the theory of NP-Completness.* Freeman, 1979 (page 65)

[Gelernter81] D. Gelernter. *A DAG-based algorithm for prevention of store and forward deadlock in packets networks.* IEEE Transactions on Communications Vol. com. 30 no 10 1981.

[Giessler78] A. Giessler, J. Hanle, A. Konig and E. Pade. *Free Buffer Allocation - An Investigation by Simulation.* Computer Networks 2 (1978) 191-208.

[Gonzalez91] N.A. Gonzalez Valenzuela. *PARX : Noyau de système pour ordinateurs massivement parallèles. - Contrôle de la communication entre processus.* Thèse Institut National Polytechnique de Grenoble, Décembre 1991.

[Gopal85] I. S. Gopal. *Prevention of Store and forward deadlock in computer networks.* IEEE Transactions on Communications Vol. com. 33 no 12 1985.

[Graham79] R.L. Graham, E.L. Lawler, J.K. Lenstra, A.H.G. Rinnoy Kan. *Optimization and approximation in deterministic sequencing and scheduling : a survey.* Annals of Discrete Mathematics pp. 287-326, North Holland 1979

[Gunther75] K.D. Gunther. *Prevention of buffer deadlocks in packet switching networks.* The IFIP-II ASA Workshop on Data Communication, Laxemburg, Austria, september 15-19, 1975.

[Gunther81] K.D. Gunther. *Prevention of deadlocks in packet-switched data transport systems.* IEEE T. COM. Vol. COM-29, no. 4, April 1981.

[Hall93] M. W. Hall, T.J. Harvey, K. Kennedy, N. McIntosh *Experiences Using the ParaScope Editor: an Interactive Parallel Programming Tool.* ACM SIGPLAN Notice vol. 28, no. 7, July 1993

[INMOS91] inmos. *The T9000 Transputer products overview manual.* inmos Databook series, first edition 1991.

[Isloor80] S.S. Isloor and T.A. Marsland. *The Deadlock Probllem: An Overview.* IEEE Computer, September 1980.

[Jesshope89a] C.R. Jesshope, P. Miller, and J. Yantchev. *High Performance Communications in Processor Networks.* ACM Computer Architecture News, Vol. 17, no. 7, June 1989.

[Jesshope89b] C.R. Jesshope, P.R. Miller, Y.T. Yantchev. *High performance communications in processor networks.* Proc. 16 th Int. Symp. on Computer Architecture. IEEE Computer Society Press, 1989.

[Jesshope91] C.R. Jesshope. *High Performance Communications Architectures.* Parallel and Disributed Processing, K. Boyanov (Ed.), 1991

[Kelly89] P. Kelly. *Functional programming for loosely-coupled multiprocessors.* Research monographs in parallel and distributed computing, the MIT press, Cambriodge Massachuset, 1989

[Kennedy80] K. Kennedy. *Automatic translation of FORTAN programs to vector forms.* Technical Report 476-029-4, Rice University, Houston, October 1980

[Kermani79] P. Kermani and L. Kleinrock. *Virtual cut-through : A new computer communication switching technique.* Computers Networks, no 3, 1979.

[Kleinrock77] L. Kleinrock and F. Kamoun. *Hierarchical Routing for Large Networks : Performance evaluation and optimization.* Computer Networks 1 (1977) 155-174.

[Langue91] Y.B. Langué Tsobgny. *PARX : Architecture de Noyau de Système d'Exploitation Parallèle.* Thèse Institut National Polytechnique de Grenoble, Décembre 1991.

[May90] D. May, P. Thompson. *Transputers and Routers: Components for Concurrent Machines* Draft paper, April 1990.

[May91] D. May. *The Next Generation of Transputers and Beyond* Proc. EDMCC2, LNCS 487, A. Bode(ed.), 1991.

[May93] D. May, P.W. Thompson, P.H. Welch. (eds.) *Networks, Routers, and Transputers : Function, Performance, and Applications* Transputers and OCCAM engineering Series, IOS Press 1993.

[Merlin80] P. Merlin and P. Schweitzer. *Deadlock avoidance in Store and Forward networks.* IEEE Transactions on Communications Vol com 28 no 3 1980.

[Miller90] P.R. Miller, J.T. Yantchev. *Developping Powerful Communication Mechanisms for Distributed Memory Computers from Simple and Efficient Message-Routing.* Proc. 5th Distributed Memory Computing Conference, Charleston USA, 1990.

[Miller91] P.R. Miller, C.R. Jesshope, Y.T. Yantchev. *The Mad-Postman Network Chip.* Proc. Transputing'91, P. Welch et al. Eds., IOS 1991.

[Mugwaneza90] L. Mugwaneza, T. Muntean and I. Sakho. *A deadlock free routing algorithm with network size independent buffering space* LNCS 457, H. Burkhart(ed.) pp. 490-501.

[Neiman69] V. I. Neiman. et commandes optimales de réseaux de connexion sans blocage. Annales des Télécom, Juillet-Août 1969.

[Nicole88] D. A. Nicole, E. K. Lloyd, J. S. Ward. *Switching networks for transputer links* Proceedings of the 8th OUG, edited by Jon Kerridge, IOS 1988, Amsterdam, Springfield, VA.

[Occam2] INMOS Ltd, *OCCAM2 Reference manual.* Prentice Hall International Series in Computer Science, 1988.

[Quillan77] J.M. Mc. Quillan, D.C. Walden. *The ARPA Network Design Decisions.* Computer Networks 1 (1977), pp. 213-289.

[Quillan80] J.M. Mc. Quillan, I.Richer, E. Rosen. *The New Routing Algorithm of the ARPANET.* IEEE T. COM. vol. Com-28, no 5, May 1980.

[Paull62] M. C. Paull. *Reswitching of connection networks.* Bell System Journal, Vol 41, 1962, pp 833 - 855.

[Peleg89] D. Peleg and E. Upfal. *A Trade-Off between Space and Efficiency for Routing Tables.* JACM vol. 36, no. 3, July 1989, pp. 510-530.

[Ramanujan73] H. R. Ramanujam. *Decomposition of Permutation Networks.* IEEE Trans. on Comp., Vol c-22, no 7 July 1973, pp 639-643.

[Roscoe87] A. W. Roscoe. *Routing messages through networks : an exercise in deadlock avoidance.* 7th O. U. G. and International Workshop on Parallel Programming of Transputers based machines, Grenoble Setpt 14-16 1987.

[Sakarovitch77] Sakarovitch. *Techniques Mathématiques de la Recherche Opérationnelle II : Eléments de la théorie des graphes* Notes de cours, Université Scientifique et Médicale de Grenoble - ENSIMAG,1977

[Sakho93] I. Sakho, L. Mugwaneza, Y. Langue. *Routing with Compact Routing Tables: ILS for Generalized Meshes.* RR. 93-7, Juin 1993, Dept. Informatique, Ecole Nationale Supérieure des Mines de Saint Etienne.

[Santoro82] N. Santoro and R. Khatib. *Routing without routing tables.* Tech. Rep.SCS-TR-6, school of Computer Science, Carleton University, Ottawa 1982.

[Santoro85] N. Santoro and R. Khatib. *Labelling and Implicit Routing in Networks.* The Computer Journal, Vol. 28, no. 1, 1985.

[Sarkar89a] V. Sarkar. *Determining Average Program Execution Times and their Variance.* ACM SIGLPAN Notices vol. 24, no. 7, July 1989 pp 298-312

[Sarkar89b] V. Sarkar. *Partitionning and Scheduling Parallel Programs for Multiprocessors.* Research monographs in parallel and distributed computing, The MIT press, Cambriodge Massachuset, 1989

[Sunshine77] C.A. Sunshine. *Interconnection of Computer Networks.* Computer Networks 1, (1977) pp 175-195 .

[Talbi93] E.G. Talbi. *Allocation de processus sur les architectures parallèles à mémoire distribuée* Thèse Institut National Polytechnique de Grenoble, Mai 1993.

[Tawbi89] N. Tawbi, P. Feautrier. *Parallélisation automatique de programmes pour ordinateur multiprocesseur à mémoire partagée.* Rapport de Recherche no. MASI 279, Université de Paris 6, Avril 1989.

[Tsaowu74] N. T. Tsao-Wu. *On Neiman's algorithm for the control of rearrangeable switching networks.* IEEE Trans on Computers, Vol c-22, no 6 June 1974, pp 737 - 742.

[Trew91] A. Trew and G. Wilson. *Past, Present, Parallel : A Survey of Available Parallel Computing Systems.* Springer Verlag, 1991.

[vLeeuwen83] J. Van Leeuwen and R. B. Tan. *Routing with Compact Routing Tables.* RUU-CS-83-16, University of Utrecht, November 1983.

[vLeeuwen86] J. Van Leeuwen and R. B. Tan. *Computer Networks with Compact Routing Tables.* In The book of L (pp. 259-273), G. Rozemberg and A. Salomaa (eds.), Springer-Verlag 1986

[vLeeuwen87] J. Van Leeuwen and R. B. Tan. *Interval Routing.* The Computer Journal, Vol. 30, no. 4, 1987.

[Waille90] P. Waille, T. Muntean. *Introduction à l'architecture des machines supernodes.* La lettrre du Transputer no 7. 1990.

[Waksman68] A. Waksman. *A permutation network.* Journal of ACM, Vol 15 no 1 January 1968, pp. 159-163.

[Wang89] C.J. Wang, V.P. Nelson and C.H. Wu. *Performance Modeling of the Modified Mesh-Connected Parallel Computer.* Proc. of the 9th Int. Conf. on Distributed Computing, Newport Beach, Calfornia 1989.

[Ward84] M. O. Ward and D. J. Romero. *Assigning parallel executable intercommunicating subtasks to processors.*

[Whobrey88] Whobrey et al.. *A Communication chip for multiprocessors.* Proc. CONPAR88, Cambridge University press, 1988.

[Wittie81] L.D. Wittie. *Communication Structures for Large Networks of Microcomputers.* IEEE Trans. on Computers, vol. C-30, no 4, pp. 284-273, April 1981.

[Wolfe89] M. Wolfe. *Optimizing Supercompilers for Supercomputers.* Research monographs in parallel and distributed computing, The MIT press, Cambriodge Massachuset, 1989